Étude génomique de l'interférence entre la réplication et la transcription comme source du stress réplicatif

Ismaël Padioleau

THÈSE POUR OBTENIR LE GRADE DE DOCTEUR DE L'UNIVERSITÉ DE MONTPELLIER

Spécialité : bioinformatique

École doctorale : CBS2, Science Chimique et Biologie pour la Santé

Unité de recherche : Institut de Génétique Humaine, CNRS-UMR9002

Étude génomique de l'interférence entre la réplication et la transcription comme source du stress réplicatif

Présentée par Ismaël Padioleau

Le 24 novembre 2017

Sous la direction de Philippe Pasero

Devant le jury composé de

Dr Domenico Maiorano, directeur de recherche, Institut de Génétique Humaine	Président du jury
Dr Jean-Charles Cadoret, enseignant chercheur, Institut Jacques Monod	Rapporteur
Dr Chun-Long Chen, directeur de recherche, Institut Curie	Rapporteur
Dr Philippe Pasero, directeur de recherche, Institut de Génétique Humaine	Directeur de thèse

UNIVERSITÉ DE MONTPELLIER

« Le hasard ne favorise l'invention que pour des esprits préparés aux découvertes par de patientes études et de persévérants efforts. »

Louis Pasteur

Remerciements

Je tiens à remercier les membres de mon jury d'avoir accepté de participer à ma soutenance, et plus particulièrement Chun-Long Chen et Jean-Charles Cadoret qui ont relu et évalué mon manuscrit. Domenico Maiorano d'avoir accepté de présider mon jury.

Merci aussi à Jean-Christophe Andrau et Hervé Seitz pour les conseils qu'ils m'ont donné au cours de mes comités de thèses.

Je remercie sincèrement Philippe Pasero, d'abord en tant que directeur de l'équipe « Maintien de l'intégrité du génome au cours de la réplication », de m'avoir fait confiance pour la prise en charge des données produites par son laboratoire. Puis pour m'avoir proposé un sujet de thèse et son accompagnement tout le long du projet. Je garderai un très bon souvenir des réunions du lundi matin, aussi bien pour les échanges techniques que pour le moment de convivialité qui les précédait.

J'aimerais remercier également tous les membres de l'équipe Pasero : Alexy, Anne-Lyne, Antoine, Armelle, Axel, Benjamin, Flavie, Hélène, Julie, Maria, Mélanie, Romain, Yea-Lih. Je passe de très bons moments au laboratoire depuis quatre ans. C'est agréable de venir travailler dans un environnement serein où la qualité scientifique est au rendez-vous. Les repas du midi et les cafés sur la passerelle me manqueront, un vrai moment de détente durant lequel on peut aborder tous les sujets, c'est important pour décompresser.

Le projet R-loop, c'est aussi celui d'Alexy et de Yea-Lih, j'apprécie la confiance qu'ils m'ont donnée pour l'analyse bioinformatique des données. Je remercie plus particulièrement Yea-Lih, pour son aide précieuse au cours de la rédaction du manuscrit ; Merci pour les nombreuses heures que tu y as consacrées. Merci surtout pour le soutien moral, ce n'est pas toujours facile de me garder à flot, mais tu as parfaitement géré la situation.

Un grand merci aussi à Romain qui a pris le relais pour l'analyse des données et l'accompagnement en bioinformatique des autres biologistes, le temps de la rédaction de ma thèse. Sans ton aide, tout aurait été plus compliqué.

Un merci tout spécial à celle qui me fait voyager, qui me pousse à aller de l'avant, qui illumine mes journées et sans qui cette page, ce manuscrit et ma vie n'auraient pas la qualité qu'ils ont aujourd'hui. Merci Nelly !

Enfin merci à ma famille qui me soutient toujours, de loin certes, mais ça, c'est surtout de ma faute.

Table des matières

Principales abreviations

γH2AX : Variant d'histone H2A phosphoryle sur la serine 319

ADN : acide désoxyribonucléique (DNA)

ARN : acide ribonucléique (RNA)

ARNm : ARN messager

ARNr : ARN ribosomique

ARNt : ARN de transfert

ASF : Facteur d'épissage alternative (pour Alternative Splicing Factor)

ASF/SF2 : Alternative Splicing Factor / Splicing Factor 2

CDC : protéine de contrôle de la division cellulaire (pour Cell-division control (protein))

CDK : Kinase dependant des cyclines

CFS : Sites Fragiles Communs

ChIP : Immunoprécipitation de la chromatine (pour Chromatin ImmunoPrecipitation)

CTD : Carboxy-terminal domain

DDK : Kinase dependant de Dbf4

DRIP : DNA/RNA Immunprecipitation

DSB : Cassure double brin (pour Double Strand Break)

ERFS : Early Replicating Fragile Site

FACS: Cytometire en flux (pour Fluorescence-activated cell sorting)

GRO-Seq : Global Run-On sequencing

IF : Immunoflorescence

IP : Immunprecipitation

mRNP : messenger RiboNucleoprotein Particle

NGS: Sequencage haut-debit (pour Next Generation Sequencing)

OK-Seq : Séquençage des fragments d'Okazaki

ORC : Complexe de reconnaissance d'origine (pour Origin Recognition Complex)

Pol II : ARN polymérase 2

Pre-IC : Complexe de pre-initiation

Pre-RC : Complexe pre-replicatif

pRPA-S33 : RPA32 phosphorylé sur la serine 33

RNase : Ribonucléase

RPA : Protéine de réplication A

RPKM : Reads Per Kilobase per Million mapped reads

shRNA : Petits ARN en épingle à cheveux (pour short hairpin RNA)

ssDNA : ADN simple brin

TOP1 : Topoisomerase 1

TSS : Site d'initiation de la transcription

TTS : Site de terminaison de la transcription

Introduction

1. Les conflits entre réplication et transcription de l'ADN

La réplication et la transcription sont deux processus génétiques fondamentaux réalisés par des complexes macromoléculaires comprenant plusieurs dizaines de protéines qui se déplacent le long de la molécule d'ADN et peuvent donc entrer en collision. Je vais brièvement présenter dans ce chapitre ces deux processus génétiques, les conséquences de leur interférence et les mécanismes cellulaires permettant de prévenir ces conflits.

1.1. La réplication de l'ADN

La réplication est le mécanisme par lequel une cellule duplique son ADN afin de transmettre une copie de son matériel génétique à ses deux cellules filles lors de la division cellulaire. Il est important que l'information passée soit identique à celle contenue dans la cellule mère car toute erreur transmise le sera également aux générations suivantes. Les mutations et les réarrangements chromosomiques induits par une réplication incorrecte ou partielle du génome contribuent au développement du cancer chez les eucaryotes supérieurs.

Dès la fin de la mitose et lors de la phase G_1 du cycle cellulaire, un complexe protéique appelé complexe pré-réplicatif (pre-RC) se forme sur l'ADN au niveau de régions particulières du génome appelées origines de réplication (Bell 2002, Coster et al. 2014). Chez tous les eucaryotes, l'assemblage du pre-RC est un processus séquentiel impliquant le complexe ORC (Origin Recognition Complex), les protéines CDC6 et CDT1 et le complexe MCM2-7 (Maiorano et al. 2004, Diffley 2004, Barlow et Nussenzweig 2014) (Fig. 1). Chez les eucaryotes supérieurs, plusieurs milliers de pre-RCs s'assemblent, sur le génome, au niveau des origines avant le début de la phase S (Aladjem 2007). Dans les cellules humaines, contrairement à la levure *S. cerevisiae*, ORC ne reconnait pas une séquence particulière sur l'ADN, mais sa fixation dépend d'une combinaison complexe de motifs nucléotidiques et de marques épigénétiques de la chromatine (Symeonidou *et al.* 2012). Après l'entrée en phase S, l'activation des kinases CDK et DDK permet le recrutement d'autres protéines au niveau du pre-RC pour former le pre-IC (pre-Initiation Complex), qui va permettre l'ouverture de la double hélice d'ADN et la formation de deux complexes de réplication appelés réplisomes (Barlow et Nussenzweig 2014). Ces derniers progressent ensuite le long de l'ADN de manière bi-directionelle au niveau de structures appelées fourches de réplication. Chaque réplisome réplique plusieurs dizaines de kilobases d'ADN à la vitesse de 1 à 2 kb/min (Huberman et Riggs 1968, Blumenthal *et al.* 1974) jusqu'à la rencontre d'un réplisome progressant en sens inverse depuis une origine adjacente, ce qui conduit à la terminaison de la réplication.

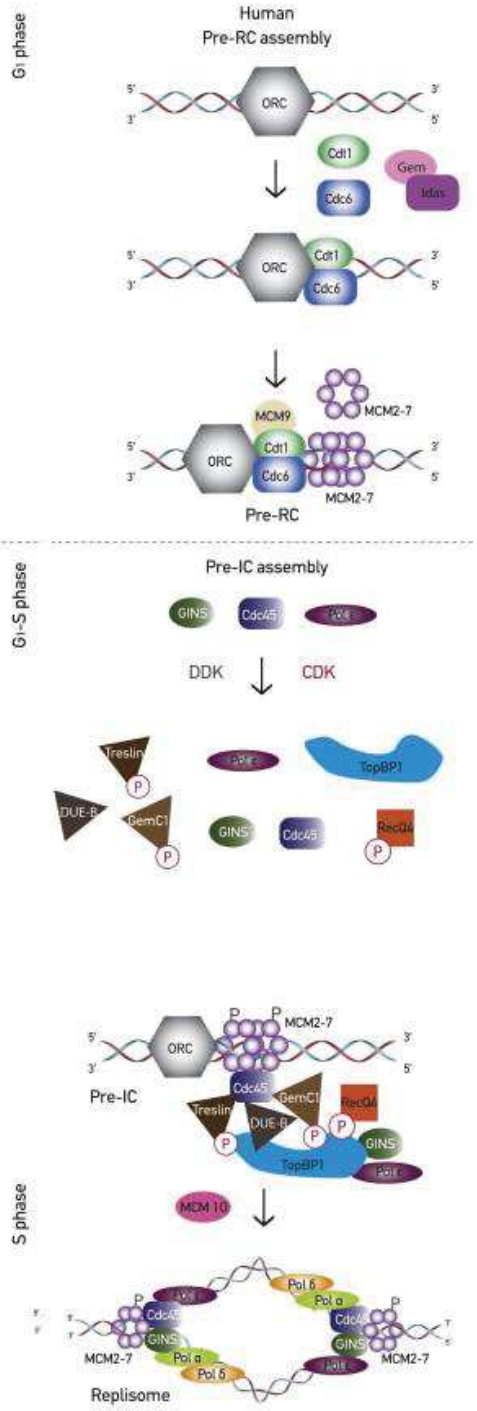


Figure 1 : Etapes successives de l'assemblage d'un réplisome à une origine de réplication. Après fixation du complexe ORC sur l'ADN, CDC6 et CDT1 sont recrutées pour faciliter le chargement de l'hélicase MCM2-7 (pre-RC). CDC45, MCM10 et le complexe GIN s'associent à MCM2-7 pour former une hélicase complète. Les protéines de régulation TopBP1 et Treslin se fixent sur le complexe alors prêt pour l'initiation de la réplication (pre-IC). CDC45 favorise l'ouverture de l'ADN et le recrutement des derniers éléments du réplisome (PCNA, Pol ε et Pol δ) (Symeonidou et al. 2012)

Le réplisome est un assemblage complexe de plusieurs types d'enzymes, telles que des hélicases et des ADN polymérases, ainsi que des facteurs accessoires assurant la coordination et la processivité de ces activités enzymatiques afin d'assurer une réplication fidèle de l'ADN. Au cours de ce processus, la séparation des brins parentaux de l'ADN est assurée par le complexe hélicase CMG (CDC45-MCM-GINS) (Moyer et al. 2006), donnant naissance à la fourche de réplication. Les deux fourches progressant à partir d'une même origine définissent un œil de réplication. En amont de la fourche, la topoisomérase I (TOP1) se charge d'éliminer les supertours positifs de l'ADN induits par l'ouverture de la double hélice afin de faciliter la réplication (Champoux, 2001).

Chez la levure, le brin orienté 3'-5' est répliqué par l'ADN polymérase epsilon (Pol ε). Il est appelé brin précoce (leading strand) car il est synthétisé de façon continue avec l'avancée de la fourche. Le brin opposé est appelé brin tardif (lagging strand), et il est généré de façon discontinue par l'ADN polymérase delta (Pol δ) (Fig. 2). En effet, les polymérases synthétisent l'ADN dans le sens 5'-3' à partir d'une amorce d'ADN-ARN (Bell 2006). La synthèse du brin tardif dépend donc de cycles successifs de synthèse d'une amorce d'ADN et d'ARN (12 nt) par l'ADN polymérase alpha-primase et de son extension par la Pol δ (~200 nt). Ces segments sont appelés fragments d'Okazaki, du nom des deux chercheurs ayant mis en évidence leur existence (Okazaki et al. 1968). Lorsqu'elle atteint l'amorce suivante, la Pol δ cède la place à la protéine FEN1 (Flap endonucléase 1) qui retire les ribonucléotides de l'amorce et permet à la Pol δ d'assurer la jonction entre les fragments d'Okazaki, avec l'aide de l'ADN ligase 1 (Zheng et Shen 2011; Stith et al. 2008). Cette situation laisse des portions d'ADN simple découvertes. Le RPA, un complexe protéique composé de trois sous-unités (RPA1, RPA2 et RPA3), se lie à l'ADN simple brin, ce qui le maintien sous cette forme et également le protège d'agressions extérieurs. La synthèse des brins précoces et tardifs doit être étroitement coordonnée avec la progression de l'hélicase afin de maintenir l'intégrité de la fourche.

La terminaison de la réplication, correspondant à la rencontre entre deux fourches, est un processus encore mal connu. Chez les bactéries et les levures, ce processus peut avoir lieu au niveau de séquences spécifiques, appelées barrières de réplication (Weiss et al. 1981, Ivessa et al. 2000). Chez l'homme, peu de barrières ont été décrites et la terminaison a généralement lieu à une position aléatoire entre deux origines (Huvet et al. 2007, Petryk et al. 2016). Du fait de la difficulté d'observer ces rencontres, les mécanismes impliqués sont encore méconnus.

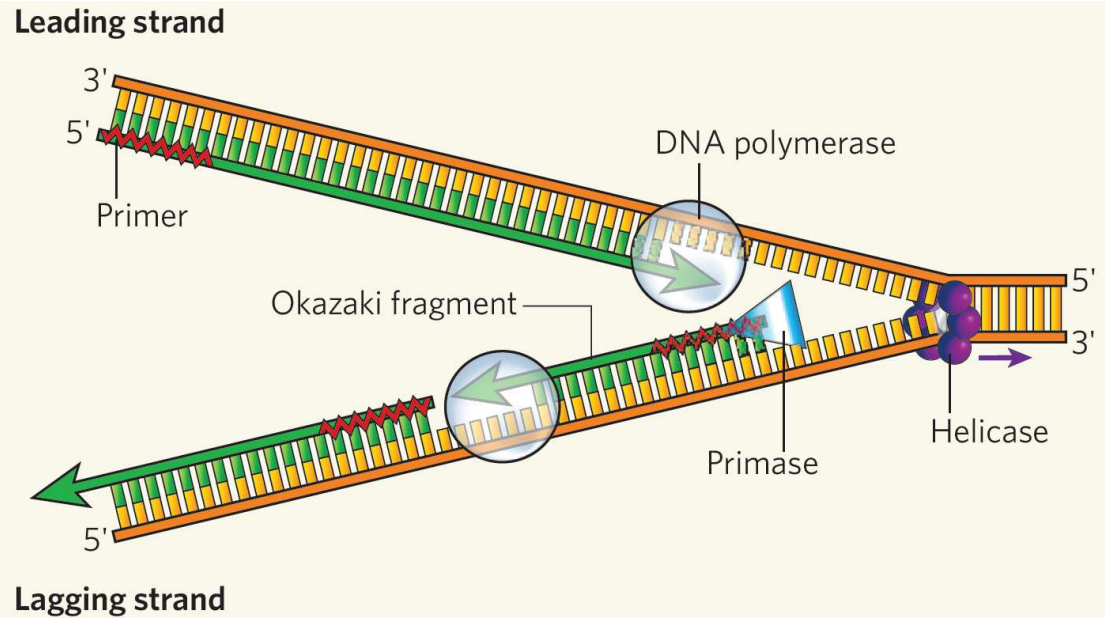


Figure 2 : Réplication de l'ADN. L'hélicase ouvre l'ADN double brin. La primase crée de courtes amorces d'ARN sur l'ADN pour initier la synthèse de l'AND par les polymérases. La réplication se faisant uniquement dans le sens 5'-3', le brin précoce est répliqué de façon continue et le brin tardif de façon discontinue. Sur le brin tardif, les fragments d'Okazaki servent de d'amorces à la polymérase (Bell 2006).

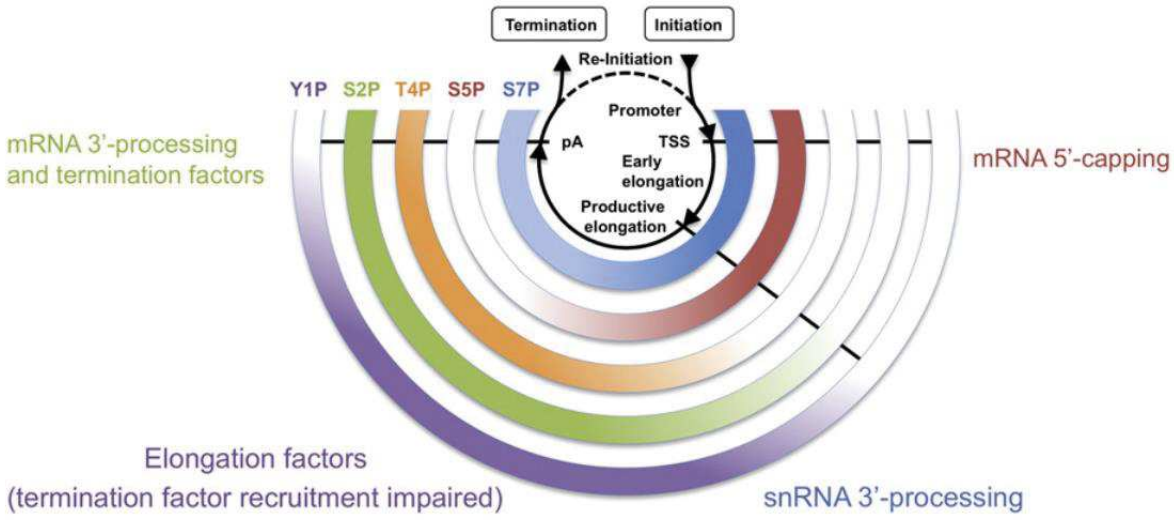


Figure 3 : Le « code CTD » pour la coordination de la transcription. Le niveau de phosphorylation des différents acides aminés du CTD code pour la progression de la polymérase à travers les différentes étapes de la transcription (Heidemann et al. 2013).

1.2. La transcription

L'expression simultanée de milliers de gènes dans le noyau de la cellule eucaryote est un processus fortement contrôlé. Il est divisé en trois étapes (initiation, élongation et terminaison) dont l'exécution dépend de nombreux facteurs. La transcription est effectuée par les ARN polymérases. Il en existe plusieurs appelées Pol I, Pol II et Pol III. Chacune est responsable de l'expression de transcrits particuliers. Pol I est active dans la transcription des ARN ribosomiques (ARNr), Pol II des ARN messagers (ARNm) et de certains ARN non-codants, et Pol III des ARNt, de l'ARNr 5S et de petits ARN non-codants.

L'ARN polymérase II est un complexe protéique formé de 12 sous-unités (Rpb1 à Rpb12) (Cramer 2004). Rpb1 est la plus grosse sous-unité et possède une extrémité C-terminale (CTD, carboxy-terminal domain), qui comprend jusqu'à 52 répétitions de l'heptapeptide Tyr-Ser-Pro-Thr-Ser-Pro-Ser, lequel peut être phosphorylé à de nombreuses positions. Le CTD joue un rôle primordial dans l'exécution de la transcription, son niveau de phosphorylation sert de checkpoint pour la transition entre les différentes étapes de la transcription (Fig. 3) (Heidemann et al. 2013).

1.2.1. L'initiation

Le processus commence par la formation d'un complexe de pré-initiation (PIC) composé de la Pol II et des facteurs généraux de transcription (TFIIA, B, C, D, E, Fet H) (Reese 2003). Le positionnement de celui-ci est extrêmement dépendant de la séquence d'ADN. La première étape est donc la reconnaissance d'une séquence promotrice, dont l'exemple le plus connu est la TATA box (Lifton et al. 1978), par les facteurs de transcription. Dans le cas de la TATA box, c'est la partie TBP (TATA-Binding Protein) du facteur TFIID qui se fixe en premier à l'ADN (Parker & Topol 1984; Buratowski et al. 1988), il va ensuite recruter les autres éléments du PIC à cette position (Fig. 4). Il existe d'autres séquences promotrices, telles que le BRE (B recognition element), et le promoteur d'un gène peut en contenir plusieurs qui seront activées par différentes voies de signalisation cellulaire (Lefstin et Yamamoto 1998). La Pol II est hypophosphorylée lorsqu'elle est recrutée (Dahmus 1996, Lin et al. 2002). Elle se fixe à l'ADN à l'aide d'une mâchoire actionnée par le facteur TFIIE (Leuther et al. 1996). Finalement, le dégagement du site promoteur est initié par la phosphorylation du CTD par le facteur TFIIH. D'autres complexes protéiques se fixent au niveau des séquences promotrices.

C'est le cas de médiateur qui favorise notamment l'action de TFIIH sur la polymérase (Kim et al. 1994, Näär et al. 2002).

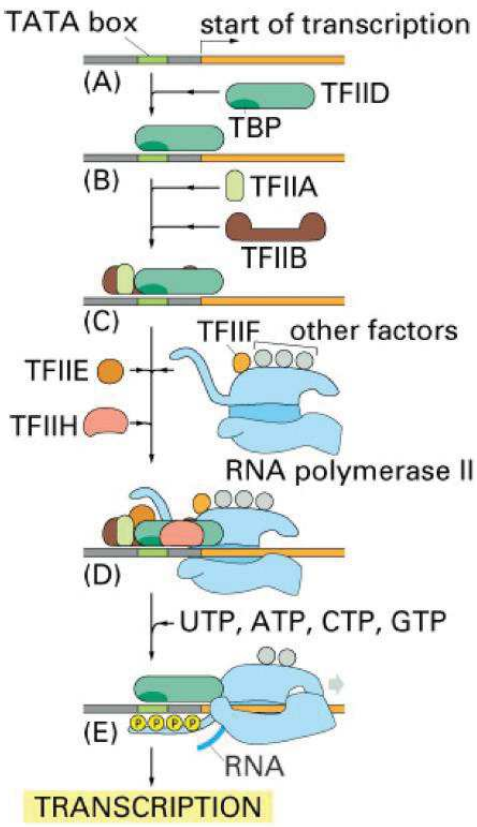


Figure 4 : Initiation de la transcription. (A,B) Le facteur de transcription TFIID reconnait la séquence TATA box et s'y fixe.(C) Les autres facteurs de transcription sont recrutés à cette position (D) ainsi que l'ARN polymérase elle-même. (E) TFIIH ouvre la double hélice d'ADN permettant à la transcription de démarrer (Alberini 2009).

1.2.2. L'élongation

L'hyperphosphorylation du CTD est responsable de l'entrée en élongation (Lin et al. 2002, Dahmus 1996). Cependant, peu après son départ du promoteur, la polymérase effectue une pause induite par les complexes NELF et SPT4/SPT5 (Yamaguchi et al. 1999, Wada et al. 1998). Cet arrêt n'est pas décrit pour tous les gènes, mais il pourrait avoir un rôle important, notamment pour permettre la formation de la coiffe protégeant la partie 5' du transcrit naissant (Wen et Shatkin 1999, Pei and Shuman 2002). La phosphorylation de NELF et de la sérine 2 du CTD par la kinase CDK9 du complexe P-TEFb permet la dissociation de NELF de la polymérase (Wu et al. 2003) et le redémarrage de la transcription (Kim et Sharp 2001, Boehm et al. 2003). Le passage de l'initiation à l'élongation entraine de nombreux changements au niveau des facteurs associés à la Pol II. La majorité des facteurs de transcription se dissocient de la polymérase et sont remplacés par des facteurs d'élongation (Elongator, Tat-Sf1, SII, ELL, CSB…) (Svejstrup 2002, Conaway et al. 2000). Ils facilitent l'avancée de la polymérase sur l'ADN en prévenant les pauses de la transcription ou en

facilitant la ré-initiation de la Pol II arrêtée (Conaway et al. 2000). D'autres éléments, comme les facteurs de remodelage (SWI/SNF, HGMG14, FACT…), changent la conformation de la chromatine en favorisant la dissociation des nucléosomes de l'ADN (Conaway et al. 2000).

1.2.3. La terminaison

Contrairement à l'initiation, la terminaison de la transcription ne se fait pas à une position précise, mais dans un domaine de la partie terminale du gène. La sérine 2 du CTD est phosphorylée sur un nombre croissant de répétitions de l'heptapeptide au fur et à mesure de la progression de la transcription (Heidemann et al. 2013). La queue de la Pol II hyper-phosphorylé s'associe avec de nombreux facteurs de clivage et de polyadénylation (CPSF-73, CstF-64, senataxin…) qui synthétisent la terminaison 3' des ARNm (McCracken et al. 1997, Proudfoot 2016). La Pol II détecte son passage sur un domaine polyadénylé actif (PAS, pour poly(A) site), ce qui initie le recrutement des facteurs de terminaison (Tran et al. 2001). La Pol II effectue alors une série de pauses sur le PAS, qui pourraient être dues à des structures de la chromatine (Alén et al. 2002) ou à l'hybridation du transcrit naissant sur l'ADN (R-loop). Cette dernière hypothèse est notamment soutenue par le rôle de certaines protéines comme senataxin, qui agissent à la fois à la terminaison de la transcription et dans la résolution des R-loops (Skourti-Stathaki et al. 2011). Le ralentissement de la progression de Pol II favorise la fixation des facteurs de terminaison (Aguilera et Garcia-Muse 2012, Skourti-Stathaki et Proudfoot 2014) et la libération progressive de la polymérase de l'ADN (Zhang et al. 2013). Finalement le CTD est déphosphorylé par FCP1, ce qui favorise le clivage de l'ARN néosynthétisé et le recyclage de la polymérase qui est redirigée vers le promoteur pour commencer un nouveau cycle (Proudfoot et al. 2002, Kamada et al. 2003).

1.3. Coordination entre réplication et transcription

Les complexes transcriptionnels parcourent en permanence une grande partie du génome, ce qui présente des avantages pour le maintien de l'intégrité du génome dans les cellules quiescentes. En effet, ceci permet de détecter rapidement les lésions de l'ADN et d'y remédier par un mécanisme, appelé réparation de l'ADN, couplé à la transcription (TCR, Mellon et Hanawalt et al. 1989). Par contre, lors de la phase S, les complexes de transcription représentent une menace pour la réplication en provoquant des collisions avec les réplisomes, ce qui constitue une source majeure de stress et d'instabilité génomique (Prado et Aguilera 2005 ; Tuduri et al. 2009 ; Barlow et al. 2014). En effet, les fourches de réplication sont des structures fragiles, dont le blocage peut induire des cassures double-brin (DSB) de l'ADN et

des réarrangements chromosomiques (délétions, inversions, duplications…) contribuant à l'apparition de tumeurs (Helmrich et al. 2011, Zeman et Cimprich 2014).

Pour éviter ces conflits, différents mécanismes ont été mis en place au cours de l'évolution. Chez la bactérie *Bacillus subtilis*, qui présente un chromosome circulaire et une seule origine de réplication, la transcription de la majorité des gènes est orientée de façon codirectionnelle avec la réplication (Kunst et al. 1997). Cette organisation du génome permet de réduire la fréquence de collisions frontales, qui sont particulièrement délétères (Azvolinsky et al. 2009, Srivatsan et al. 2010). Chez les eucaryotes, la situation est plus complexe car le génome est organisé en plusieurs chromosomes linéaires, portant chacun un grand nombre d'origines de réplication. Chez les vertébrés, ces origines sont généralement peu efficaces et le sens de réplication d'une région donnée peut varier d'une cellule à l'autre au sein d'une population, ce qui rend les conflits avec la transcription inévitables. La co-orientation de la transcription et de la réplication a néanmoins été mise en évidence pour certaines régions du génome humain (Huvet et al. 2007, Petrik et al. 2016).

Une autre manière de limiter les conflits entre réplication et transcription est de séparer ces événements spatialement et temporellement. Chez les eucaryotes, la transcription a lieu pendant tout le cycle cellulaire, alors que la réplication est restreinte à la phase S. La transcription est globalement active en phase S, mais elle a lieu au sein de territoires nucléaires distincts par rapport à la réplication (Wei et al. 1998, Helmrich et al. 2013). C'est le cas du gène codant pour la beta-globine, fortement exprimé dans les cellules érythroïdes (Vieira et al. 2004), ou du gène codant pour l'histone H4, présent en plusieurs copies sous l'influence de différentes séquences régulatrices (Holmes et al. 2005). Le contrôle du moment d'activation des origines suivant un programme de réplication très précis (Jackson 1995), permet aussi de limiter les conflits en séparant temporellement la transcription et la réplication (Gilbert et al. 2002). Dans la majorité des cas, les gènes actifs sont répliqués précocement (Woodfine et al. 2004). Cependant, les gènes codant pour des facteurs de réplication sont transcrits en début de phase S et répliqués tardivement (Meryet-Figuiere et al. 2014).

Finalement, il est important de noter que les origines actives sont souvent proches des éléments de régulation de la transcription et que l'activité de la transcription peut influencer l'activation des origines (Jackson et al. 1998, Knott et al. 2009). En effet, l'ouverture de la chromatine, l'acétylation des histones, la méthylation de l'ADN, les facteurs de transcription et certaines propriétés des séquences aux promoteurs comme les ilots CpG favorisent à la fois

l'expression des gènes et l'assemblage des pre-IC (Fig. 5) (Hiratani et al. 2009, Cadoret et al. 2008, Danis et al. 2004). L'impact de cette coordination sur les conflits réplication-transcription reste cependant mal connu.

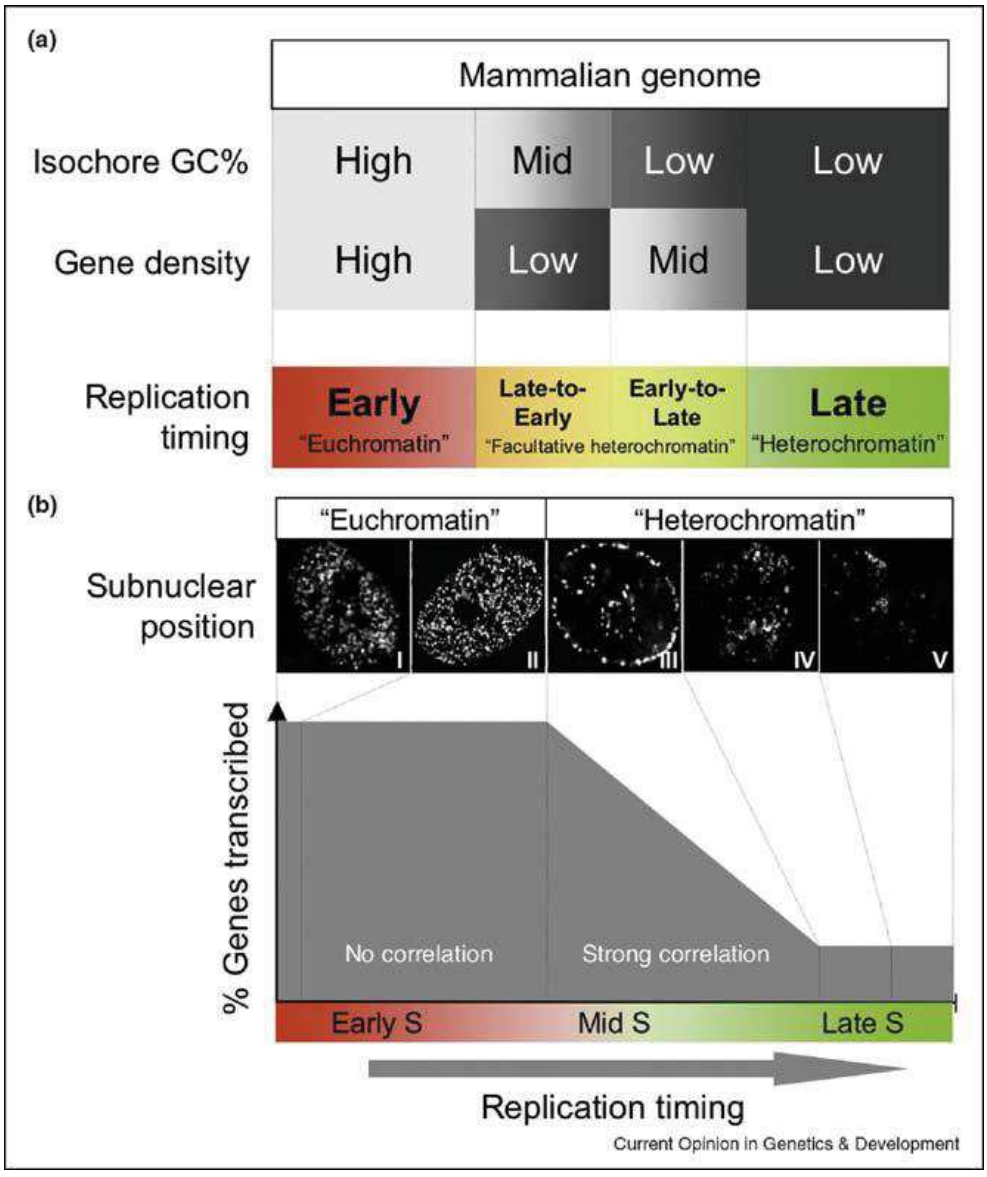


Figure 5 : Relation entre le timing de réplication, le GC%, la position sub-nucléaire et la transcription. (a) Les régions riches en gènes et au GC% élevé sont répliquées précocement, alors que celles pauvres en gènes et au GC% faible sont répliquées tardivement. Le timing de réplication des rares zones où ces deux propriétés s'inversent est variable selon le type cellulaire. (b) Les zone riches en gènes et positionnées à l'intérieur du noyau sont répliquées précocement. A l'inverse, les zones pauvres en gènes et positionnées en périphérie du noyau sont répliquées tardivement. (Hiratani et al. 2009).

1.4. Les réponses aux conflits réplication-transcription

Malgré l'existence de différents mécanismes permettant de limiter les collisions entre réplication et transcription, ces conflits sont inévitables, notamment dans le contexte de cellules tumorales ou prétumorales, présentant une dérégulation de voies oncogéniques et une prolifération aberrante (Halazonetis et al. 2008). Dans ces cellules, le ralentissement ou le blocage des fourches de réplication, suivant un processus appelé stress réplicatif, représente une menace pour l'intégrité du génome et nécessite l'intervention des voies de protection du génome. La signalisation des lésions de l'ADN peut être prise en charge par trois voies, ATM (ataxia-telangliectasia mutated), ATR (ataxia-telangliectasia mutated and RAD3 related) ou DNA-PK (DNA-dependente protein kinase) (Blackford et Jackson 2017), trois protéines kinases qui organisent la réponse au stress et la transduction du signal en activant à leur tour les kinases CHK1 et CHK2 (Checkpoint Kinase 1 et 2). Les protéines ATM et DNA-PK interviennent en réponse aux cassures de l'ADN alors qu'ATR signale le blocage des fourches (Fig. 6). Dans le contexte de conflits entre réplication et transcription, il est probable que la réponse initiale soit prise en charge par ATR (García-Muse et Aguilera 2016).

Le recrutement d'ATR aux sites de stress se fait par l'intermédiaire d'ATRIP (ATR-interacting protein, Cortez et al. 2001). Cette protéine interagit avec RPA (Replication Protein A), un complexe se fixant à l'ADN simple brin. En effet, le ralentissement de la fourche causé par un obstacle provoque généralement le découplage des polymérases et de l'hélicase, ce qui entraîne l'accumulation d'ADN simple-brin (ssDNA), le recrutement de RPA (Heller et al. 2006) et la fixation d'ATRIP. Une fois présent aux fourches bloquées, ATR active la kinase effectrice CHK1, ce qui permet de phosphoryler un très grand nombre de substrats dans la cellule. Il a été montré que la nature de ces cibles et la réponse au stress dépendent du niveau de la menace (Koundrioukoff et al. 2013). En effet, ATR est capable de phosphoryler un nombre limité de facteurs au niveau de la fourche, mais si le stress persiste, cette réponse locale est amplifiée pour atteindre d'autres cibles via l'activation de CHK1. Parmi ces cibles, on trouve notamment les origines de réplication plus tardives, autours de la zone problèmatique, dont l'activation est inhibée par CHK1 en réponse au stress réplicatif (Kumar et al. 2009, Luciani et al. 2004).

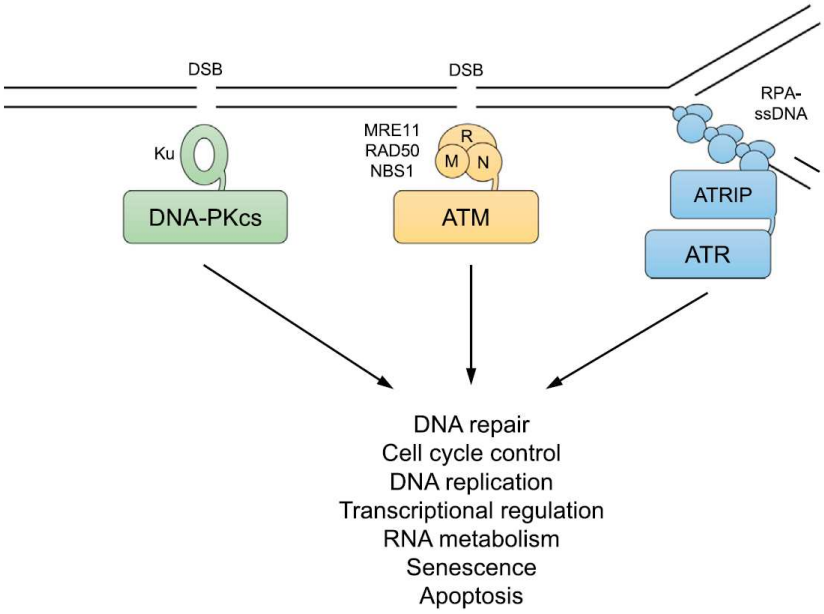


Figure 6 : Les voies de signalisation des lésions à l'ADN. DNA-PK est recruté par la protéine Ku liée aux cassures double-brin. ATM est activé par le complexe MRN (MRE11-RAD50-NBS1). ATR est recruté par ATRIP au niveau de l'ADN simple brin recouvert par RPA (Blackford et Jackson 2017).

En réponse à un stress réplicatif modéré, l'accumulation de ssDNA et la phosphorylation de CHK1 sont indétectables, mais ATR est recruté et phosphoryle RPA sur la sérine 33 (Dungrawala et al. 2015, Técher et al. 2017). RPA est également phosphorylée par ATM sur d'autres résidus en réponse aux cassures de l'ADN, mais on considère que la phosphorylation sur la sérine 33 est une marque spécifique du stress de réplication (Vassin et al. 2009).

La fourche de réplication est particulièrement vulnérable quand elle est arrêtée, surtout s'il y a accumulation d'ADN simple brin. Pour protéger l'intégrité du génome, le réplisome doit être stabilisé, c'est-à-dire que ses différents composants doivent rester physiquement associés avec la fourche. Il a été démontré qu'en cas de stress réplicatif et en absence d'ATR, plusieurs composants du réplisome ne sont plus détectables au niveau des fourches de réplication (Trenz et al. 2006, Ragland et al. 2013, Hashimoto et al. 2012).

L'arrêt de la fourche peut aussi entrainer sa réversion, un processus au cours duquel l'appariement des deux brins nouvellement synthétisés permet de stabiliser la structure. Cette configuration protège les brins naissants contre une dégradation trop étendue et favorise la réactivation de la fourche (Neelsen et Lopes 2015). Dans cette situation, ATR contrôle l'activité des CDKs (Cyclin-Dependent Kinase), WRN (Werner syndrom ATP-dependent helicase) et SMARCAL1 afin de prévenir une résection excessive des brins naissants (Sorensen et Syljuasen 2012, Couch et al. 2013, Ammazzalorso et al. 2010). Ce contrôle est

important car la dégradation des brins naissants augmente l'exposition des brins parentaux, le recrutement de RPA et l'activation des voies de réponses au stress.

Si la fourche reste trop longtemps arrêtée ou si elle casse, certaines protéines de la voie FANC (Anémie de Fanconi) comme FANCD2 et FANCI sont également phosphorylées par ATR (Lossaint et al. 2013, Sirbu et al. 2013). BRCA2 et FANCD2 recrutent RAD51, qui protège les brins naissants contre une résection excessive par MRE11 en formant un nucléofilament (Schlacher et al. 2011, Schlacher et al. 2012). BRCA1 et BRCA2 seraient également impliqués dans la résolution des collisions, principalement quand celles-ci impliquent des R-loops (Bhatia et al. 2014, García-Rubio et al. 2015).

Enfin, il est important de noter qu'une des cibles principales d'ATR est un variant d'histone appelé H2AX. La forme phosphorylée de H2AX, appelée γ-H2AX, est détectée en réponse au stress réplicatif. H2AX est également phosphorylé par ATM en réponse aux cassures double brin de l'ADN (Fig. 7). Contrairement à la forme phosphorylée de RPA sur la sérine 33, qui n'est détectée qu'à proximité des fourches bloquées, γ-H2AX s'étend sur plusieurs centaines de kb à partir du site de cassure (Rogakou et al. 1999).

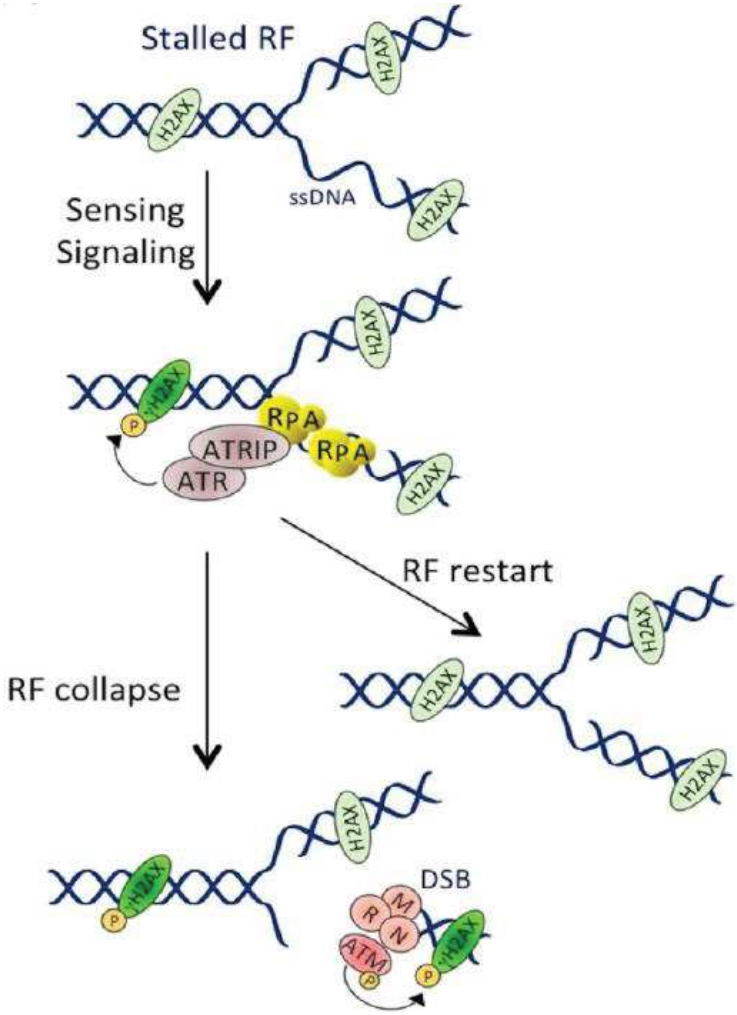


Figure 7 : Voie de réponse au stress réplicatif par ATR. L'arrêt de la fourche induit la formation d'ADN simple-brin, rapidement recouvert par RPA. ATRIP reconnait cette structure et entraine le recrutement et l'activation d'ATR. ATR phosphoryle de nombreux substrats sur place, dont RPA et H2AX. La fourche peut redémarrer si le stress est transitoire ou s'écrouler si le problème perdure, ATM sera alors recruté et entrainera une deuxième vague de phosphorylation de H2AX (Adapté de Bezine et al. 2014).

2. Les R-loops

Les R-loops sont des hybrides ADN-ARN dont la structure est connue depuis 1976 (Thomas et al. 1976). Un ARN dont la séquence est complémentaire à un brin d'ADN peut, sous certaines conditions, envahir le brin d'ADN et former une structure à trois brins : un hybride ADN/ARN et un ADN simple brin (single-strand DNA, ssDNA). Thomas a décrit les conditions favorisant la formation des R-loops *in vitro*, à forte concentration de formamide et à une température où la double hélice d'ADN se sépare en ADN simple brin. Des propriétés qui, dès l'année suivante, seront utilisées pour établir une méthode permettant d'identifier les séquences d'ADN ayant une homologie de séquence avec un ARN : le R-loop mapping (White et al. 1977). Les R-loops auront alors leur premier moment de gloire avec une quinzaine de publications par an au début des années 1980 (Fig. 8).

C'est seulement en 1995 que la preuve de la formation de R-loop *in vivo* sera établie (Drolet et al 1995). Les R-loops sont vues comme une structure rare, qui se forme de manière opportuniste dans des conditions spécifiques, ici l'absence de topoisomérase I (TOP1). Depuis, l'étude des R-loops *in vivo* a permis de déterminer leur importance dans de nombreux processus cellulaires et d'identifier les conditions favorisant leur formation. Ces découvertes se sont accélérées avec le développement des techniques de séquençage à la fin des années 2000, notamment grâce au DRIP-Seq (cf. Chapitre sur les techniques de séquençage), qui combine l'immunoprécipitation des R-loops grâce à un anticorps monoclonal reconnaissant spécifiquement les hybrides ARN/ADN (Boguslawski et al. 1986), et une technique de séquençage à haut débit (Ginno et al. 2012).

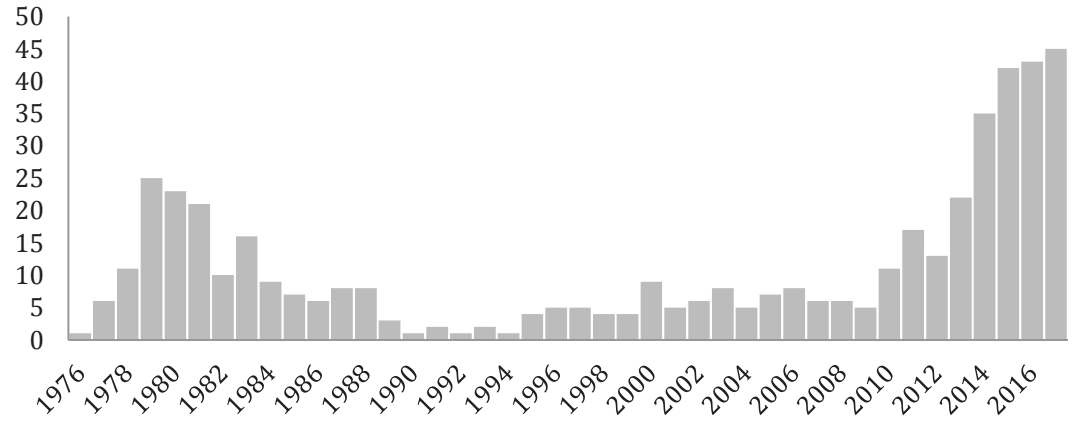


Figure 8 : Nombre de publications parlant des R-loops par année (Données générées par PubMed).

2.1. Les R-loops dans la cellule

Les conditions exactes de la formation des R-loops ne sont pas encore parfaitement connues. Le scénario le plus probable est l'hybridation de l'ARN naissant sur son ADN complémentaire après le passage de l'ARN polymérase (« Thread back model », Roy et al. 2008). L'ADN derrière la polymérase accumule les supertours négatifs, ce qui favorise la séparation des deux brins et l'invasion par le brin d'ARN. D'autres conditions favorisent l'assemblage et la stabilité de l'hybride (Fig. 9), notamment la richesse en guanine du brin non transcrit (Roy et al. 2009), la présence d'un « nick » sur l'ADN (Roy et al. 2010) ou encore la formation de G-quartet sur l'ADN simple brin transcrit (Duquette et al. 2004).

La localisation des R-loops sur le génome grâce aux méthodes de séquençage a permis de proposer plusieurs hypothèses sur le rôle qu'elles jouent dans la cellule. Ce sont notamment les parties promotrices et terminatrices des gènes qui sont le plus riches en R-loops.

Chez l'homme, les promoteurs de gènes sont riches en cytosine, notamment dans les ilots CpG, qui servent à la régulation de leur expression. En effet, la méthylation des cytosines dans ces sites est un signal d'inactivation de la transcription (Tate and Bird 1993). La présence d'un fort GC skew (déséquilibre du ratio G/C) en amont de ces sites favorise aussi la formation des R-loops (Ginno et al. 2013). Il a été proposé que la formation de R-loops au promoteur protège les cytosines de l'action des ADN méthyletransférases, favorisant ainsi l'expression des gènes en empêchant leur inactivation (Ginno et al. 2012).

De même, la séquence ADN suivant la partie poly-(A) en 3' de certains gènes est riche en G. Ces séquences G-riche favorisent la formation des R-loops ce qui a pour effet de ralentir l'ARNP II (Pol II). Cette situation permet l'intervention de la senataxine (SETX), une hélicase qui aide à détacher l'ARN et permet à l'exonucléase Xrn2 de se fixer au site de clivage de la queue poly-(A) et de libérer Pol II (Skourti-Stathaki et al. 2011). Ce modèle est proposé pour les zones riches en gènes où l'arrêt de la Pol II doit se faire sur une courte distance (Ginno et al. 2013). Ce modèle a depuis été complexifié, il est envisagé que la formation des R-loops en fin de gène favorise la transcription antisens et la formation d'ARN double brin. Cette structure atypique entraine la di-méthylation, par RNAi, des histones H3K9 (H3K9me2), signal de condensation de la chromatine. L'hétérochromatine ainsi formé stoppe l'ARN Pol II et favorise la terminaison de la transcription (Skourti-Stathaki et al. 2014).

De nombreux autres rôles sont proposés pour les R-loops, le mieux décrit étant leur rôle dans la commutation isotypique des immunoglobulines chez les lymphocytes B activés (Yu et

al. 2003). Mais elles permettraient aussi de maintenir l'ADN centromérique condensé (Nakama et al. 2012) et de contrôler l'expression de certains gènes en se formant lors de l'expression d'ARN non codant (Sun et al. 2013, Boque-Sastre, 2015).

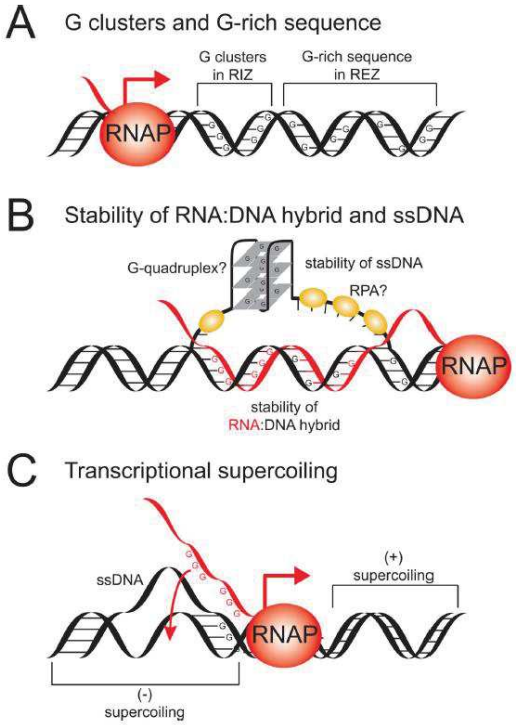


Figure 9 : Les facteurs favorisant la formation des R-loops. (A) Les séquences riche en Guanine favorisent la formation et l'élongation des R-loops (RIZ : R-loops Initiation Zone, REZ : R-loops Elongation Zone). (B) La formation et la stabilité des R-loop peuvent-être induites par la stabilisation du brin simple libre, par la fixation de protéines ou par la formation de structures secondaires. (C) Le surenroulement négatif de l'ADN derrière l'ARN polymérase favorise l'hybridation de l'ARN sur l'ADN (Hamperl et Cimprich 2014).

2.2. Prévention et résolution des R-loops

Comme nous l'avons vu, les R-loops interviennent dans de nombreux processus et sont présentes sur une grande partie du génome, environ 5% du génome et 50% des gènes (Ginno et al. 2012). Il est donc important pour la cellule d'avoir les outils pour contrôler la formation et la dégradation des R-loops.

La prise en charge de l'ARN à sa sortie de la polymérase est un important mécanisme de protection. De multiples protéines intervenant sur l'ARN naissant ont été identifiées comme ayant une influence sur la formation des R-loops (Santos-Pereira et Aguilera, 2015). Entre

autres, le complexe THO/TREX qui associe la maturation de l'ARN et son export hors du noyau chez la levure (Huertas et al. 2003), ou ASF/SF2, un facteur d'épissage qui charge les mRNP (messenger RiboNucleoprotein Particle) sur l'ARN naissant chez les vertébrés (Li et Manley 2005). Il semble que chacun de ces facteurs joue un rôle différent car l'absence de l'un d'entre eux favorise la formation d'hybrides et n'est pas compensée par les autres. Mais dans ces cellules défectueuses, la surexpression de RNase H1 supprime les R-loops et restaure en partie la stabilité du génome.

Les enzymes RNase H1 et H2 sont un autre mécanisme de contrôle des R-loops. Elles agissent après la formation des hybrides et ont pour fonction de dégrader la partie ARN (Ceritelli et al. 2009). Elles contiennent un site de fixation aux hybrides (HBR) et un domaine RNaseH qui catalyse la séparation de l'ARN. RNase H2 joue un rôle dans la réplication en dégradant la partie ARN des fragments d'Okazaki (Qiu et al. 1999) et élimine les ribonucléotides incorporés par erreur dans l'ADN lors de la réplication (Eder et Walder 1991). D'autres enzymes agissent sur les R-loops déjà formées, c'est le cas des hélicases DEAH box protein 9 (DHX9) (Chakraborty et Grosse 2011), DDX19 (Hodroj et al. 2017), senataxin (SETX) (Skourti-Stathaki et al. 2011) et aquarius (AQR) (De et al. 2015,).

Enfin les topoisomérases TOP1 (Tuduri et al. 2009) et TOP3B (Yang et al. 2014) préviennent la formation des R-loops en supprimant les supers tours accumulés durant la transcription et la réplication.

2.3. L'instabilité génomique due aux R-loops

Malgré tous les processus cellulaires auxquels elles participent et tous les outils de contrôle à disposition, les R-loops se sont révélées être une menace pour l'intégrité du génome.

L'étude du stress réplicatif dans des cellules accumulant les R-loops a montré que l'interférence entre la réplication et la transcription n'était pas nécessairement directe et pouvait se faire par l'intermédiaire des R-loops (Aguilera 2002, Gan et al. 2011). Dans ces cellules, la progression des fourches est ralentie, les mutations associées à la transcription s'accumulent ainsi que les DSBs. La surexpression de RNase H1 réduit l'ensemble de ces problèmes, les liant ainsi directement à la présence de R-loops problématiques.

Les ERFS (Early Replicating Fragile Sites) et les CFS (Common Fragile Sites) sont des sites de cassures spontanées liées à la réplication (Barlow et al. 2013). Les CFS se situent au

niveau de très longs gènes (>1 Mb) répliqués tardivement (Le Beau et al. 1998). Il a été montré qu'une ARN polymérase met plus de temps pour parcourir la totalité de ces gènes que la longueur du cycle cellulaire, ce qui rend l'interférence avec la réplication inévitable. De plus, comme des R-loops se forment pendant la transcription de ces gènes (Helmrich et al. 2011), il est possible que le signal de compaction de la chromatine due aux R-loops (Castellano-Pozo et al. 2013) soit la source du blocage de la réplication et des cassures qu'elles provoquent. Une fois de plus, la surexpression de RNase H1 supprime l'instabilité liée à ces régions, ce qui confirme l'implication des R-loops dans ce processus. Les ERFS sont situés dans des clusters de gènes riches en CpG là où les R-loops se forment facilement. Néanmoins, le rôle des R-loops dans l'instabilité des ERFs n'a pas encore été étudié.

3. Protéines cibles pour favoriser la formation des R-loops

Pour étudier la formation des R-loops et leur rôle dans l'apparition du stress réplicatif, nous avons utilisé deux lignées cellulaires humaines modifiées. Les résultats d'une étude précédente de notre laboratoire ont montré que la réplication était perturbée dans les cellules déplétées pour la topoisomérase 1 (TOP1) ou pour le facteur d'épissage alternatif (ASF/SF2) (Tuduri et al. 2009). Le stress réplicatif étant réduit par le traitement de ces cellules avec la RNase H1, ces données suggèrent que l'accumulation des R-loops contribue au stress réplicatif, mais cette relation n'avait pas été directement établie à l'époque, faute de technique efficace pour détecter et cartographier les R-loops.

Pour notre projet, nous avons utilisé un système inductible de shRNA ciblant TOP1 et ASF/SF2 dans des cellules HeLa. L'utilisation des cellules HeLa nous a permis par la suite de comparer nos résultats à un ensemble important de données publiées pour cette souche (Core et al. 2008, Hansen et al. 2010, The ENCODE Project Consortium 2012).

Je vais présenter ici les rôles de TOP1 et ASF dans la cellule et comment leur inhibition favorise l'étude du stress réplicatif associé à l'activité transcriptionnelle.

3.1. Topoisomérase 1

Les topoisomérases sont des enzymes capables de couper l'ADN de façon transitoire pour supprimer un stress topologique. Il en existe 6 dans la cellule humaine (TOP1, TOP1mt, TOP2α, TOP2β, TOP3α et TOP3β) et elles participent à de nombreux processus cellulaires : relâchement des surenroulements dus à la transcription et à la réplication, coupure de l'ADN

au cours de la recombinaison, condensation des chromosomes et séparation des chromatides sœurs au cours de la mitose (Champoux 2001, Wang 2002).

Elles peuvent être recrutées sur l'ADN par de nombreuses protéines (Pommier et al. 2016) et leur fixation n'est pas séquence dépendante, ce qui leur permet d'intervenir partout sur le génome (Porter et Champoux 1989, Jaxel et al. 1991). Une fois fixées, elles coupent l'ADN sur un ou deux brins et permettent sa rotation, ce qui supprime les supertours. La coupure de l'ADN par les topoisomérases utilise un mécanisme de transestérification qui est facilement réversible et limite les risques de cassure de l'ADN (Pommier et al. 2010, Vos et al. 2011, Chen et al. 2013). On sépare les topoisomérases en deux classes, selon qu'elles coupent l'ADN sur un brin (classe I) ou sur deux brins (classe II).

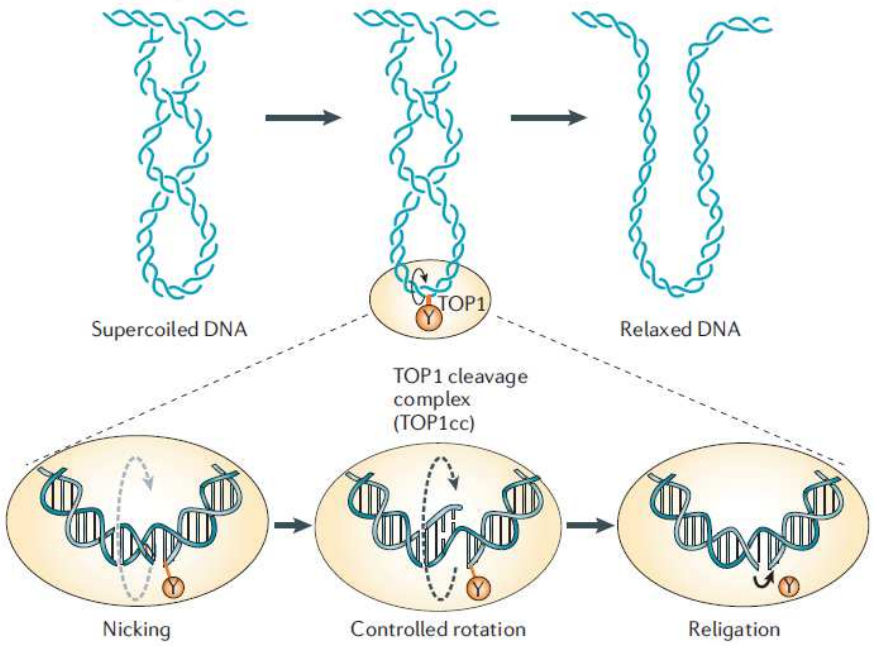


Figure 10 : Relâchement de l'ADN surenroulé par l'action de topoisomérase 1 (adaptée de Pommier 2006)

TOP1 fait partie de la classe I, elle intervient principalement en amont et en aval des polymérases (ARNP et ADNP) pour supprimer les supertours accumulés (Liu et Wang 1987, Pommier et al. 2006) (Fig. 10). Au niveau de la transcription, elle facilite notamment le recrutement du facteur de transcription TFIID au niveau de la TATA-box (Shykind et al. 1997, Sperling et al. 2011) et facilite l'initiation de certains gènes en favorisant par exemple le désassemblage des nucléosomes (Pedersen et al. 2012, Roedgaard et al. 2015). Une étude

récente montre que TOP1 est ensuite maintenue inactive, ce qui favorise l'accumulation de surenroulement et l'ouverture de l'ADN au TSS. Elle est finalement de nouveau activée par la sous-unité BRD4 de la Pol II pour redémarrer après la pause de la transcription (Baranello et al. 2016).

TOP1 intervient également dans la réplication en supprimant les supertours à l'avant de la fourche (Strumberg et al. 2000). Elle est également localisée aux origines de réplication, mais le rôle qu'elle y joue n'est pas encore élucidé (Abdurashidova et al. 2007, Falaschi 2009, Hu et al. 2009, Rampakakis et al. 2010). Les cellules où TOP1 est déplétée accumulent un stress réplicatif et des cassures de l'ADN. Notre équipe a observé dans ces cellules un ralentissement des fourches et ce phénomène pourrait être dû à l'accumulation de R-loops (Tuduri et al. 2009). Chez la levure *Saccharomyces cerevisiae,* l'activation du checkpoint dépendant de Rad53 en réponse aux cassures de l'ADN a également été détectée en l'absence de TOP1 (Brill et al. 1987, Kim et Wang 1989, Bermejo et al. 2007). Les résultats montrent que son action prévient les rencontres de la réplication et de la transcription et l'instabilité génomique (Bermejo et al. 2007, Miao et al. 2007) (Fig. 11).

Tous ces résultats font de la topoisomérase 1 une cible intéressante dans le traitement des cellules cancéreuses, qui ont des activités très élevées de réplication et de transcription. Le Topotecan, un dérivé de la camptothécine, fixe la topoisomérase I à l'ADN et inhibe l'élongation de la transcription (Ljungman et Hanawalt 1996). Le traitement favorise la formation de R-loops et de cassures double brin (Sordet et al. 2009, Sordet et al. 2010). Les R-loops pourraient être dues à l'absence d'activité catalytique de TOP1 (Hazra et al. 2004), mais leur formation pourrait aussi être due à l'arrêt de la Pol II bloquées sur l'ADN par TOP1 (Sordet et al. 2008, Sordet et al. 2009).

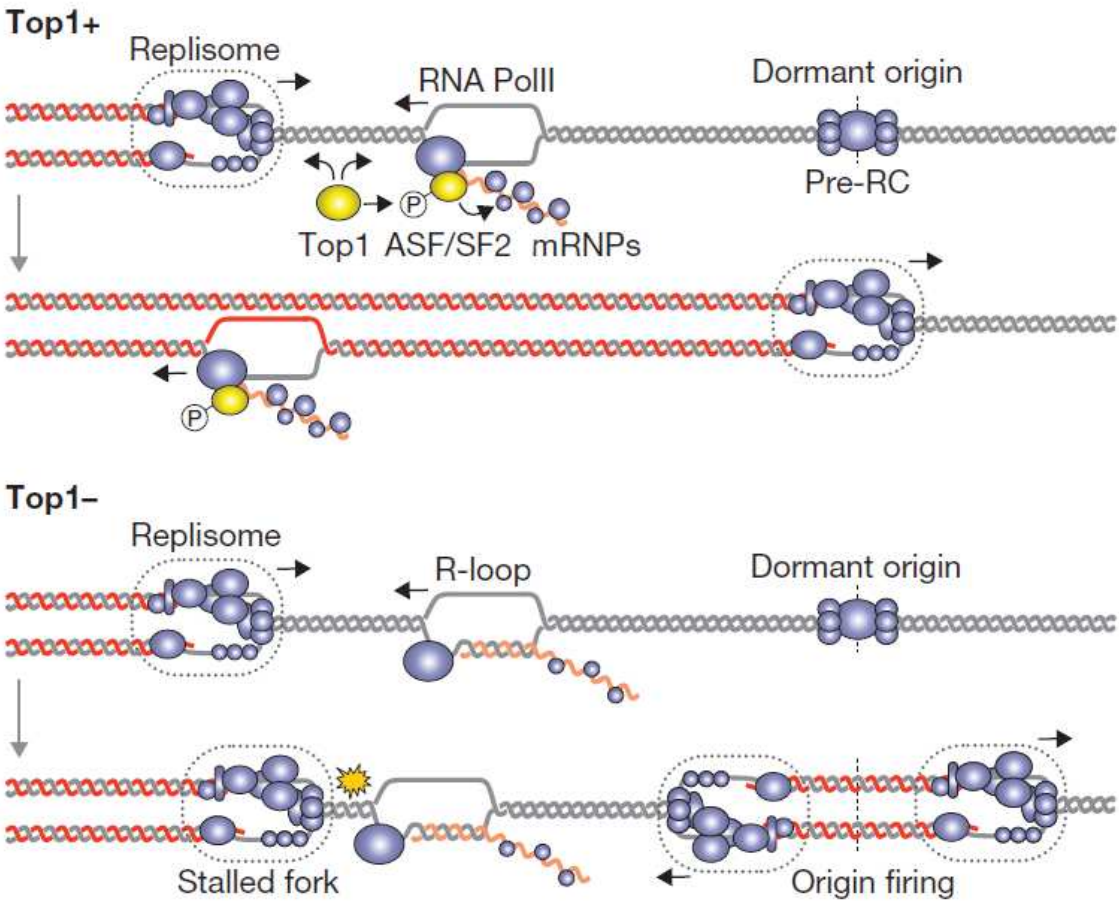


Figure 11 : Modèle de l'action de TOP1 aux sites de rencontre de la réplication et de la transcription. En plus de son rôle dans la relaxation de l'ADN, TOP1 favorise l'assemblage des mRNP par ASF/SF2. Dans une cellule normale, la fourche avance à vitesse constante et les origines dormantes sont répliquées passivement (TOP1+). En absence de TOP1 (TOP1-), les défauts de transcription favorisent la formation de R-loops et l'arrêt des fourches. Si le problème est persistant, les origines dormantes peuvent être activées pour compléter la réplication (Tuduri et al. 2009).

3.2. Le facteur d'épissage ASF/SF2

ASF/SF2 (Alternative Splicing Factor / Splicing Factor 2), aussi connue sous le nom SRSF1 (Serine and Arginine Rich Splicing Factor 1), est un membre de la famille des protéines SR (serine/arginine) qui interviennent principalement dans l'épissage constitutif et alternatif (Wang et al. 2008). ASF/SF2 a également d'autres rôles dans la prise en charge des ARN, par exemple dans l'export des ARNm (Huang et al. 2003) et leur traduction (Lemaire et al. 2002, Sanford et al. 2004).

ASF/SF2 possède deux sites N-terminaux de reconnaissance de motif ARN (RRM) et un domaine C-terminal riche en dipeptide serine/arginine (RS). La partie C-terminale est importante pour les interactions protéine-protéine favorisant l'assemblage du spliceosome et

peut servir pour la fixation au pre-ARNm (Kohtz et al. 1994, Shen et al. 2004). L'état de phosphorylation de ce domaine joue également un rôle dans l'interaction avec la protéine transportin-SR qui permet le transport d'ASF/SF2 entre le noyau et le cytoplasme (Cáceres et al. 1998, Kataoka et al.1999). En effet, ASF/SF2 reste associée à l'ARNm quand celui-ci est exporté dans le cytoplasme. Dans sa forme hyper-phosphorylée, ASF/SF2 s'associe au pre-ARNm par son site RRM au niveau des séquences ESE/ISE (Exon/intronic Splicing Enhancer) et promeut l'assemblage du spliceosome (Misteli et al. 1998, Shen et Green 2004). La protéine est ensuite déphosphorylée par la protéine phosphatase I (PPI) et facilite le transport de l'ARNm dans le cytoplasme (Novoyatleva et al. 2008).

ASF/SF2 intervient dans l'épissage alternatif de centaines de gènes (Anczuków et al. 2015), notamment des régulateurs de l'apoptose (BCL-xS, CASP9) (Massiello et Chalfant 2006, Leu et al. 2012). Sa surexpression contribue à l'épissage alternatif des formes anti-apoptiques de ces gènes et favorise la cancérogénèse (Karni et al. 2007). Au contraire, son inhibition entraine l'instabilité génomique, l'arrêt de la cellule en G2 et l'apoptose (Li et Manley 2005). Dans les cellules de poulet DT40 et dans les cellules humaines HeLa, l'instabilité génomique résultant de la déplétion d'ASF/SF2 a été liée à l'accumulation de R-loops (Li et Manley 2005, Tuduri et al. 2009). En effet, en plus de son rôle dans le splicing des ARN, le recrutement d'ASF/SF2 sur l'ARNm empêche son hybridation sur l'ADN et la formation de R-loops.

4. Le séquençage à haut débit

L'ADN fut découvert en 1869 par Friedrich Miescher et sa structure élucidée en 1953 par Watson et Crick (Watson et Crick 1953), mais les technologies permettant de décrypter la séquence de l'ADN ont été développées beaucoup plus tard. Dans un premier temps, les chimistes furent capables de déterminer la composition en nucléotides d'un acide nucléique, mais pas l'ordre dans lequel ceux-ci apparaissent (Holley et al. 1961). Puis les premières séquences ordonnées furent générées, d'abord celle des tRNA (Holley et al. 1965, Cory et al. 1968, Dube et al. 1968), puis le premier gène codant pour une protéine (Min-Jou et al. 1972) et enfin le premier génome (Bacteriophage MS2, 3.5 Kb) en 1976 (Fiers et al. 1976). C'est à ce moment-là qu'une première révolution eut lieu, le séquençage par la méthode de Sanger (Sanger et al. 1977). Bien qu'utilisant des techniques similaires aux méthodes précédentes, cette méthode était plus précise, plus rapide et plus facile à utiliser. A travers de nombreuses améliorations, elle a ouvert la porte à l'automatisation et à la création des premières machines

de séquençage. Le remplacement de la radioactivité par la fluorescence donnera naissance à la première génération de séquenceurs commerciaux, dont le premier modèle fut mis sur le marché par Applied Biosystems Inc. (ABI) en 1986 (Model 370A DNA Sequencing System). Ce sont d'autres machines d'ABI (ABI PRISM 3700/3730) qui seront utilisées pour le séquençage du premier génome eucaryote, celui de la levure, publié en 1997, et pour le séquençage du génome humain publié en 2004 (Clayton et al. 1997, Lander et al 2004). Ces projets, qui ont coûté des milliards de dollars, ont nécessité la collaboration de centaines de laboratoires et pris plus d'une dizaine d'années. L'année suivant la publication du premier génome humain aura lieu une nouvelle révolution. En 2005, est commercialisé le premier séquenceur de deuxième génération ou séquençage à très haut débit. Cette innovation va faire chuter drastiquement le prix du séquençage (Fig. 12) et initier l'ère du –Seq (RNA-Seq, ChIP-Seq…). Au cours des dix dernières années, les séquenceurs utilisés dans nos laboratoires et la mise en place de nombreuses techniques associées ont remplacé les anciens protocoles et révolutionné notre compréhension de la génétique. Je vais décrire ici les principes généraux du séquençage de deuxième génération et les techniques de production de données que j'ai utilisées au cours de ma thèse.

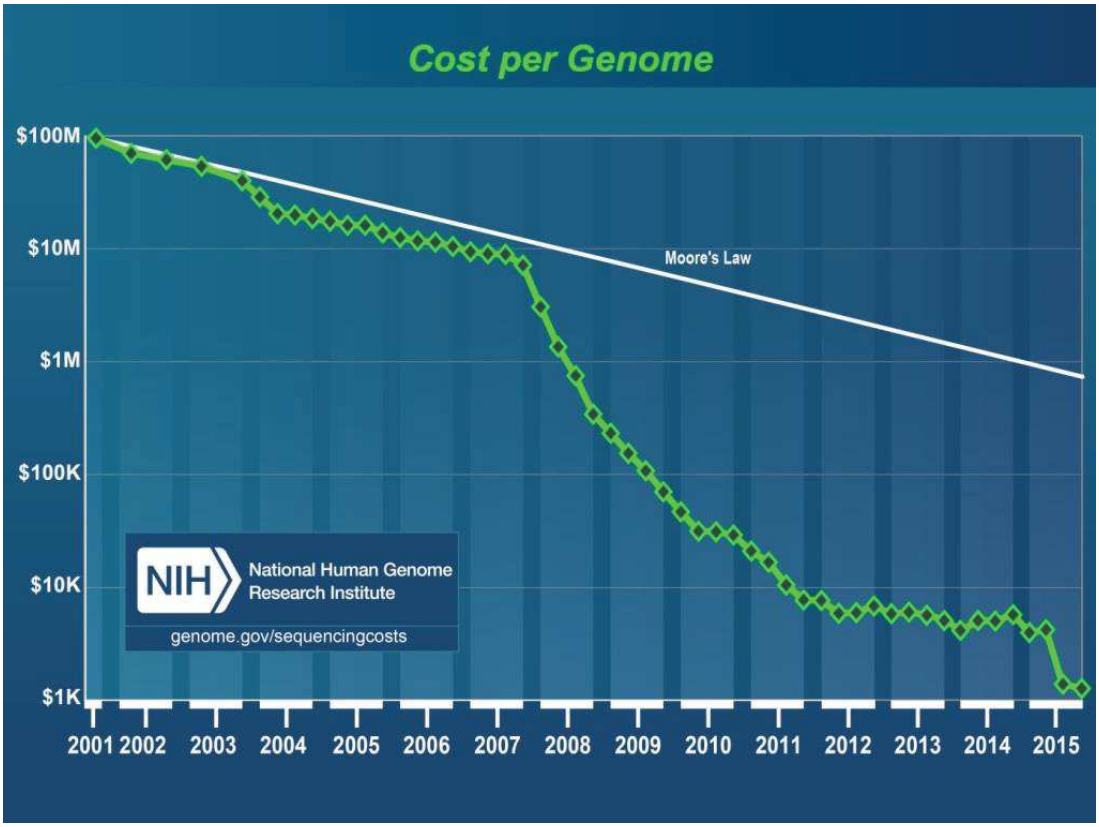


Figure 12: Evolution du prix du séquençage d'un génome humain (Data from the NHGRI Genome Sequencing Program (GSP)).

4.1. Principe

Les technologies de nouvelle génération apparues à la fin des années 2000 dominent encore le marché. Comme pour les séquenceurs de première génération, de nombreuses évolutions ont eu lieu. Même si les principales étapes que je vais présenter n'ont pas changé, l'amélioration des protocoles, des réactifs chimiques et du matériel de détection ont multiplié par 1000 la quantité d'information générée en une expérience (Fig. 13).

La première évolution apportée par les séquenceurs à haut débit est le nombre de séquences produites en parallèle. Plusieurs millions de séquences sont générées en une seule exécution, mais elles sont de petite taille (35-300 bp), contrairement à ce qui se passe avec la technologie Sanger, qui permet de générer des fragments allant jusqu'à 800 bp, mais ne permet de produire que 96 séquences à la fois. La première étape du séquençage de nouvelle génération est la constitution d'une librairie. La stratégie est simple, l'ADN d'intérêt est fragmenté en petits morceaux par sonication ou par coupure enzymatique. Des adaptateurs sont fixés aux extrémités des fragments, ils serviront à la purification et à la fixation sur la surface de séquençage. Optionnellement, une séquence de 6-7 nucléotides (index) peut être ajoutée à chaque échantillon. Cela permet de mélanger plusieurs expériences pour le séquençage (multiplexing) et de les séparer par la suite grâce à leur index unique (démultiplexing).

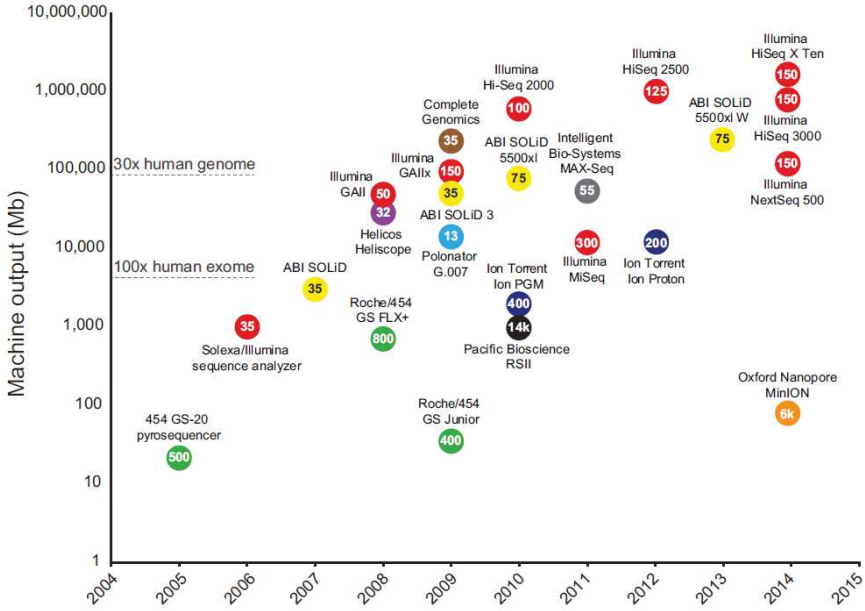


Figure 13: Evolution de la quantité de séquences produites par les machines de nouvelle génération (Reuter et al. 2015)

La librairie est ensuite chargée dans le séquenceur qui va déterminer les séquences des fragments qui la composent. Les machines distribuées par le leader du marché (Illumina) utilisent une technique appelée séquençage par terminateur réversible. Dans cette méthode, on séquence l'ADN par cycle.

Dans un premier temps une amorce est ajoutée à une extrémité du fragment, elle va permettre d'initier la synthèse de l'ADN par la polymérase. A chaque cycle, les quatre dNTPs marqués par fluorescence et associés à un terminateur, qui bloque la synthèse, sont ajoutés au milieu et le nucléotide complémentaire de la séquence est incorporé. Le milieu est ensuite nettoyé, puis le nucléotide ajouté est déterminé par un capteur de fluorescence, chaque type de base (A, C, T et G) étant associé à une couleur spécifique. Le terminateur est ensuite clivé par rayonnement ultra-violet, ce qui permet de commencer un nouveau cycle d'incorporation de base.

Bien qu'ils aient été améliorés depuis les premières versions, les systèmes optiques ne sont pas assez sensibles pour détecter la lumière émise par l'ajout d'un seul nucléotide. Pour permettre la détection du signal, il faut donc amplifier les fragments avant séquençage. Les fragments sont fixés sur un support solide (flowcell) grâce aux adaptateurs et vont être amplifiés localement, avant le séquençage, pour former un cluster contenant quelques milliers de copies de la même séquence (Fig. 14). A chaque étape de détection, toutes les séquences du cluster étant identiques, le signal envoyé est amplifié, ce qui facilite sa lecture. Lorsqu'une base est lue par la machine, un algorithme interprète le signal et accorde à la base un score – élevé, si le signal est clair, et faible, voire nul, si le signal est très mauvais (Phred score). Les machines de dernière génération produisent directement en sortie un fichier contenant les séquences ainsi que le score de confidence pour chaque base.

Les séquences enregistrées dans ces fichiers sont appelées lectures (ou reads en anglais). Le fragment peut être séquencé par une extrémité (single-end) ou par les deux (pair-end), la deuxième solution est plus chère mais offre des avantages lors de l'analyse des données (voir partie RNA-seq).

La quantité d'informations produites par ces machines et le prix, devenu abordable pour un plus grand nombre de laboratoires, ont permis à de nombreux biologistes d'intégrer l'utilisation du séquençage de nouvelle génération dans leurs projets. Dans les années qui ont suivi l'arrivée des techniques de séquençage de deuxième génération, un grand nombre de protocoles biologiques ont été créés ou adaptés à partir de méthodes préexistantes pour inclure

le séquençage (Fig. 15). Je vais présenter brièvement les techniques utilisées pour générer les données que j'ai analysées au cours de ma thèse.

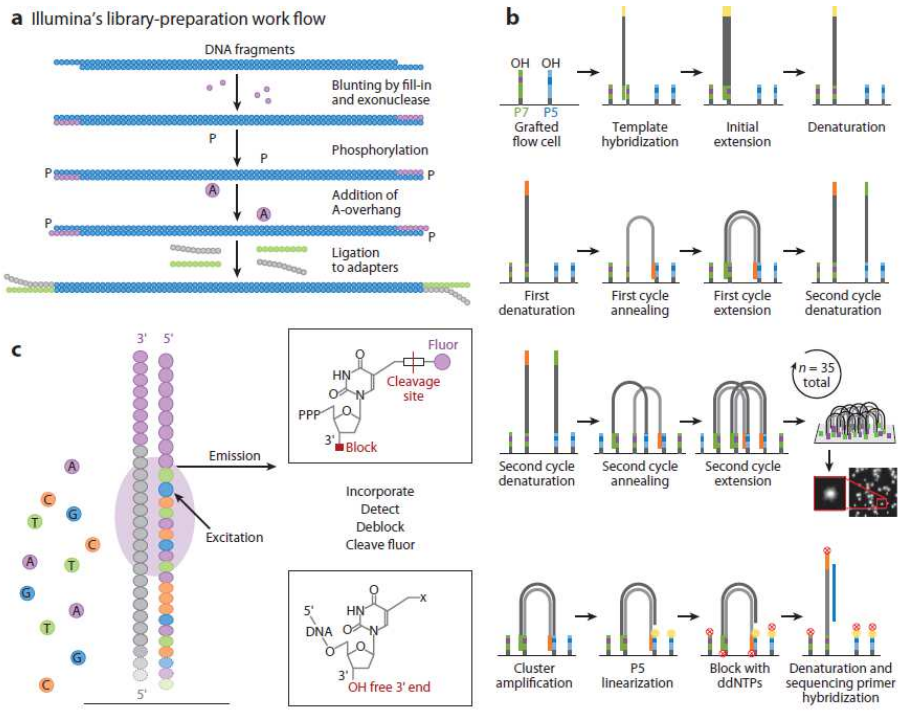


Figure14 : Protocole illumina (Madris 2013)

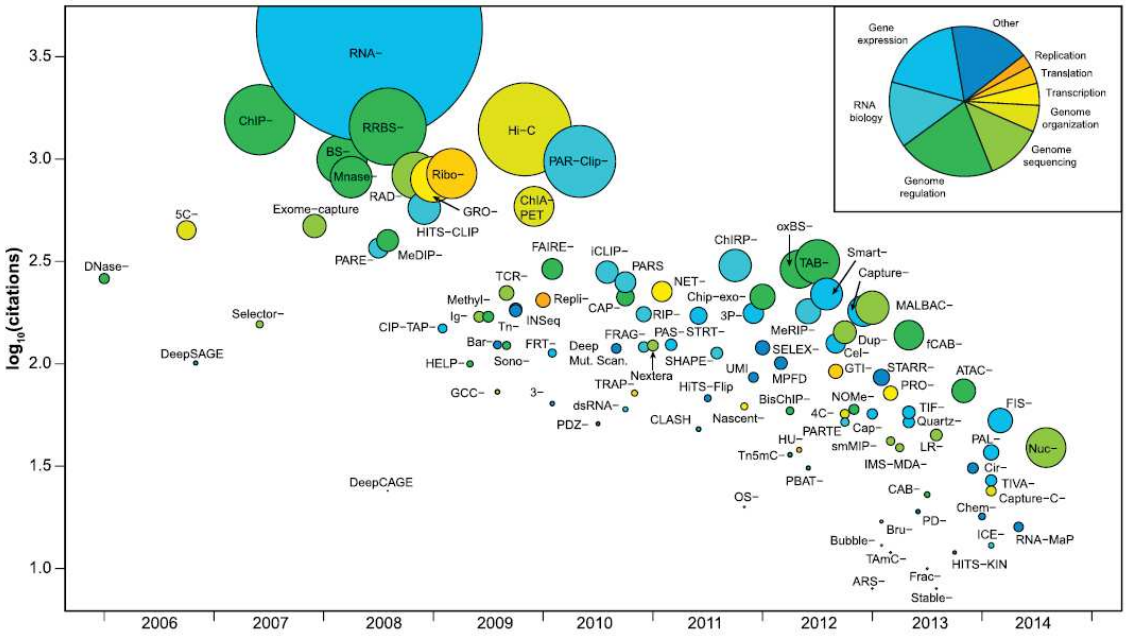


Figure 15 : Techniques de séquençages depuis l'arrivée des séquenceurs de deuxième génération (Reuter et al. 2015)

4.2. RNA-Seq

Utilisé pour la première fois en 2008 (Nagalakshmi et al. 2008) le RNA-Seq permet de déterminer la séquence de l'ensemble des ARNs extraits d'une population de cellules. Il a été développé pour identifier les gènes, leurs exons et leur position sur le génome, mais il permet, en théorie, aussi d'identifier tous les ARNs non codants produits dans un type cellulaire donné. La caractérisation de ces séquences codantes et non codantes est importante pour comprendre comment fonctionne le génome et par exemple pour étudier les séquences régulatrices associées à ces transcrits. La technique de RNA-Seq est quantitative. Elle permet de déterminer l'abondance d'un ARN donné dans la cellule, qui dépend à la fois du niveau de transcription du gène correspondant et de la stabilité de cet ARN. Le développement récent de techniques d'amplification linéaire des acides nucléiques permet aussi de séquencer les ARN issus d'une cellule unique.

4.2.1. Principe

Le protocole consiste en la capture des ARN obtenus à partir d'une population de cellules (Fig. 16). La première version utilise des oligo(dT) pour sélectionner les ARN messagers matures, grâce à leur queue poly(A). Le protocole a par la suite été adapté pour capturer d'autres types d'ARN (ARN non-codant, microARN…). La transcription inverse est ensuite utilisée pour générer des ADNc qui sont finalement fragmentés et utilisés pour le séquençage. Les séquences obtenues sont alors généralement alignées sur un génome de référence et en révèlent les portions exprimées. Les lectures peuvent aussi être assemblées *de novo* pour déterminer le transcriptome d'un organisme dont on ne dispose pas du génome.

Le RNA-Seq est une révolution dans le monde de la biologie génétiques. Avec moins de matériel génétique de départ, il obtient des résultats plus précis que les méthodes précédentes. Il permet de positionner les gènes au nucléotide près sur le génome, de connaitre les exons exprimés dans un type cellulaire et de détecter les variations de la séquence comme les SNP (Single Nucleotide Polymorphism). L'analyse du niveau d'expression des gènes est aussi bien meilleure que celle obtenue avec les anciennes technologies (microarrays).

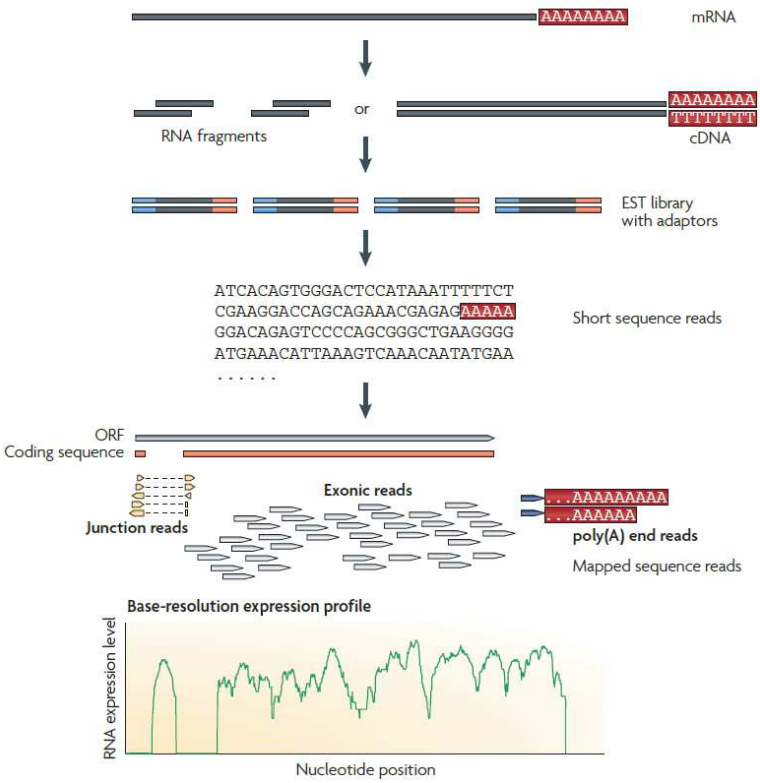


Figure 16: Description d'une expérience de RNA-Seq. Une librairie d'ADNc est créée, par transcription inverse, à partir des ARN purifiés. L'ADN obtenu est fragmenté et des adaptateurs sont fixés à une extrémité des fragments, ou aux deux si le séquençage est paired-end. La séquence des fragments est obtenue par séquençage. Le profil d'expression est généré par l'alignement des données sur la référence. En bas, le profil d'un transcrit avec un intron est montré (Wang et al. 2009).

```
@P6KDKXP1:160:C7WPTACXX:1:1101:1491:1981 1:N:0:GCCAAT
TTCAATTCTGGTATGTTTACAAAAGTGGAATGGGCTTTGTGTGAATTAGAG
+
@CCFFFEEHHHHDFHJJJCIGIEECHAHIHHEEIJJEIJH@HHBDFIHHFE
```

Figure 17 : Exemple d'encodage d'un read au format fastq.

4.2.2. L'analyse bioinformatique

Le contrôle qualité des données brutes

Le traitement de données commence par le contrôle de la qualité des séquences brutes produites par le séquenceur. Plusieurs paramètres sont importants à vérifier, comme la qualité

par base, le nombre de reads dupliqués ou la distribution des bases par position. Les séquenceurs modernes fournissent les données directement au format .fastq (Fasta with quality), une structure simple en quatre lignes devenue un standard dans le domaine (Fig. 17):

- Le nom donné par la machine au read précédé du symbole '@'
- La séquence en nucléotides codée par 5 lettres : A, C, T, G et N (indéterminé)
- Une ligne avec seulement un +. Dans les anciennes versions, le nom était répété, mais cela augmente énormément la taille du fichier pour une information inutile.
- Enfin une chaîne de caractères ASCII, de la même longueur que le read, codant pour la qualité de chaque base appelée par la machine.

Certains logiciels, comme FastQC produisent un rapport complet sur la qualité du séquençage (Andrews : https://www.bioinformatics.babraham.ac.uk/projects/fastqc/).

Si le rapport montre que la qualité des reads diminue en 3' ou qu'un k-mer (séquence de nucléotides de taille k) est très répété, un logiciel comme CutAdapt (Martin 2011) peut être utilisé pour couper une partie des lectures. Retirer les parties de moindre qualité améliore les résultats obtenus dans la suite de l'analyse.

Il est possible que l'échantillon soit contaminé par un autre organisme, le plus commun étant la contamination par le biologiste au cours de l'expérience. Ce problème est habituellement détecté après l'alignement sur la référence (voir plus loin), quand une partie importante des reads ne correspond pas au génome de l'organisme étudié. Le logiciel FastQ-screen (Wingett : https://www.bioinformatics.babraham.ac.uk/projects/fastq_screen/) propose d'aligner tout ou partie des lectures de l'échantillon sur un ensemble de références choisi par l'utilisateur pour détecter d'où vient la contamination. Si un organisme correspond à la contamination, le logiciel propose de faire l'alignement sur ce génome et de produire un fichier .fastq sans les lectures issues de cet organisme.

Une fois les reads bruts préparés, ils peuvent être utilisés pour un assemblage *de novo* ou pour un alignement sur le génome ou le transcriptome de référence. La première solution est utilisée lorsque le génome de référence de l'organisme étudié n'a pas encore été assemblé ou si de nouveaux transcrits, absents du transcriptome de référence, sont recherchés.

Alignement

L'utilisation d'un génome de référence est la solution la plus répandue, le nombre de génomes disponibles augmentant avec le temps et la majorité des organismes modèles ayant été séquencés (homme, levures, drosophile…). L'alignement ou mapping est l'opération qui consiste à retrouver sur le génome la position des reads. Il existe de nombreux logiciels pour réaliser cette étape. L'analyse comparative des performances de onze de ces logiciels a récemment permis de mettre en évidence de grandes différences (Engström et al. 2014). Certains logiciels sont optimisés pour un usage précis, comme la découverte de variants ou de jonctions d'épissages non annotées. Le choix du programme utilisé dépendra donc des besoins du projet.

Tous ces logiciels proposent de nombreuses options qui influencent la qualité de l'alignement. On peut par exemple autoriser une lecture à s'aligner sur plusieurs positions dans le génome, tolérer un gap de quelques bases dans l'alignement ou encore des différences de nucléotide entre le read et la référence. Toutes ces informations seront utilisées pour donner un score à chaque alignement qui peut permettre ensuite de filtrer les résultats.

Les logiciels d'alignement produisent généralement un fichier binaire, le BAM (Binary Alignment Map). Chaque ligne de ce format décrit un alignement : position sur le génome, qualité de l'alignement, séquence nucléique du read, position des éventuels gaps…

Assemblage

L'assemblage est utilisé quand le génome de référence n'est pas disponible ou pour recréer un transcriptome sans a priori. Le principe de l'assemblage est de reconstruire l'ARN séquencé en unifiant les reads par similarité de séquence. Le fragment obtenu par croisement de plusieurs read est appelé contig. Un bon assemblage produit un petit nombre de contigs, de grande taille et avec une couverture important en reads. Cependant, les technologies de séquençage actuelles ne se prêtent pas très bien à cet exercice. La taille des lectures 50-150 bp ne permet pas un bon chevauchement des séquences, et au lieu d'être recomposés en une séquence, les transcrits sont souvent divisés en dizaines de contigs. L'arrivée de la troisième génération de séquenceurs produisant des reads de plus grande taille pourrait régler certains des problèmes actuels.

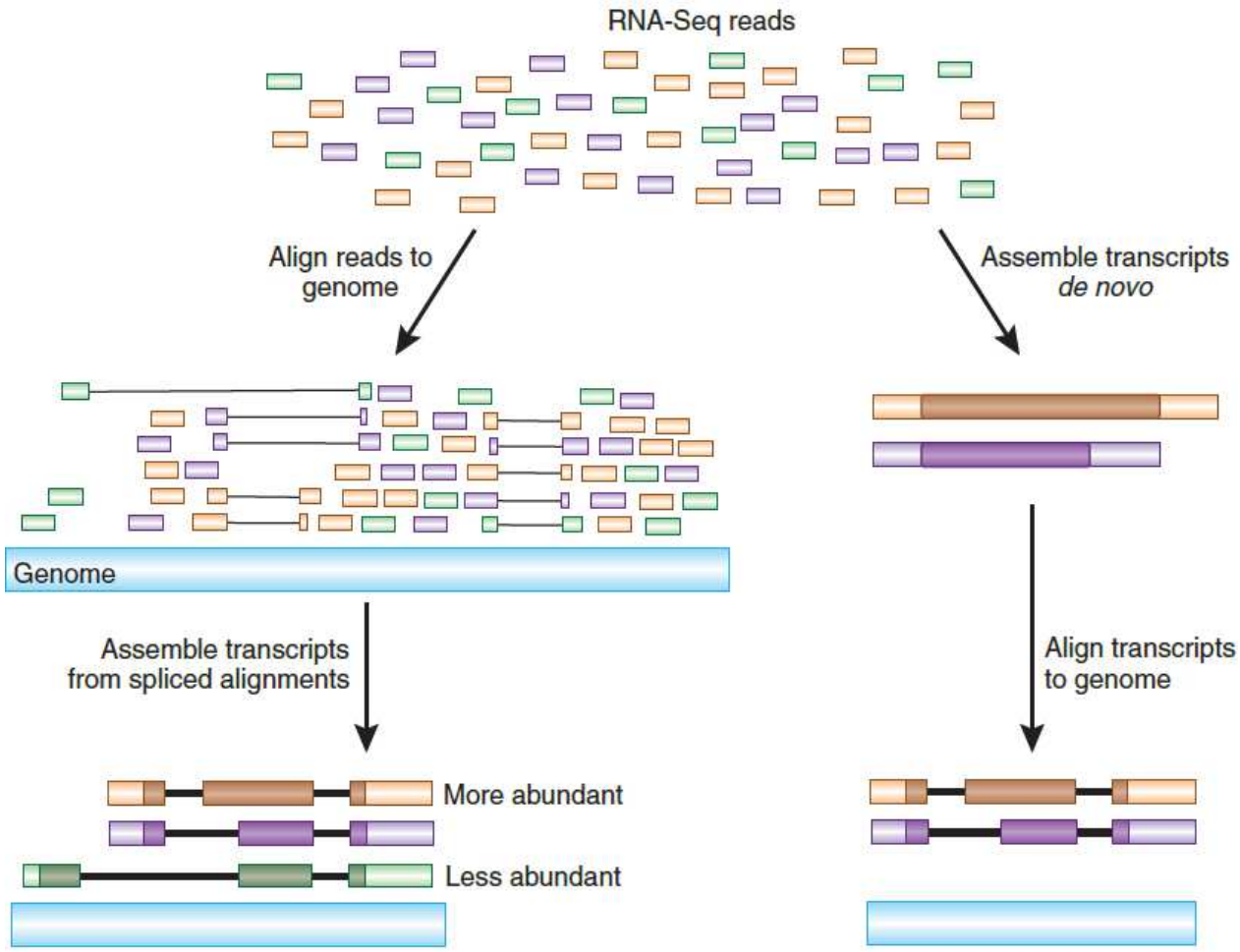


Figure 18 : Utilisation du RNA-Seq pour la découverte et la quantification des transcrits. A gauche, la stratégie par alignement sur le génome, puis assemblage grâce aux reads couvrant une jonction. A droite, la technique par assemblage, le résultat pouvant être ensuite aligné sur le transcriptome connu. A cause des difficultés de l'assemblage *de novo*, cette technique peut perdre l'information des transcrits les moins abondants (Haas et Zody 2008).

Contrôle qualité de l'alignement

Avant de passer à l'analyse proprement dite des données, il faut vérifier la qualité de l'alignement. On s'attend à avoir environ 80% de lectures alignées avec un bon score. Cette valeur varie avec le type d'expérience et la difficulté par exemple d'avoir suffisamment de matériel de départ pour créer la librairie. En dessous de 70% il peut être intéressant de chercher s'il n'y a pas eu une contamination ou si la librairie n'est pas composée d'un grand nombre de séquences répétées qui sont rejetées par le logiciel d'alignement, suite à un biais de PCR. La distribution des reads sur le génome est également importante et certains logiciels vérifient cette distribution sur le corps des gènes. Un enrichissement en 3' par exemple peut indiquer un problème dans la qualité de l'extraction des ARN. Il existe de nombreux programmes permettant de produire des rapports pour évaluer la qualité du séquençage à

partir d'un fichier BAM (Picard : http://picard.sourceforge.net, RSeQC : Wang et al. 2012 ou Qualimap : García-Alcalde et al. 2012).

Ces trois premières étapes, le contrôle qualité des données brutes, l'alignement (ou assemblage) et le contrôle qualité de l'alignement, sont appliquées à l'immense majorité des données de séquençage, quel que soit le protocole (RNA-Seq, ChIP-Seq…). C'est en quelque sorte le prétraitement qui permet ensuite d'analyser le signal.

La découverte de nouveaux transcrits

Qu'ils aient été obtenus par assemblage ou par alignement, la taille des reads fait qu'il est rare qu'une séquence couvre plusieurs exons. Mais grâce au séquençage paired-end, il est possible d'avoir les deux extrémités d'une séquence de 300 à 800 paires de bases. Si chacun des reads d'une paire est aligné sur un exon différent, c'est que ceux-ci participent à la formation d'un transcrit. Cette méthode permet de découvrir des épissages alternatifs mais aussi de quantifier différentes populations d'ARN produites par un même gène (Fig. 18).

La découverte de transcrits par assemblage est possible, mais comme décrit précédemment, la taille des reads rend cette tâche compliquée. De plus, augmenter le nombre de reads en espérant obtenir plus de chevauchement de séquences n'est pas efficace car cela augmente aussi les erreurs d'assemblage.

Quantification et expression différentielle des transcrits

La majorité des expériences de RNA-Seq sont réalisées pour évaluer le niveau d'expression des gènes dans une cellule à un temps et dans une condition donnée. Il est possible ensuite de comparer l'expression de plusieurs gènes dans une même condition ou de regarder l'effet de différentes conditions sur l'expression des gènes.

Pour cela, il faut d'abord quantifier l'expression des transcrits. Il existe des logiciels tels que HTSeq-count (Anders et al. 2015) ou featureCounts (Liao et al. 2014) qui, à partir des résultats de l'alignement et de l'annotation des gènes, comptent le nombre de lectures par annotation (gène, exon, transcrit). D'autre programmes comme Cuflinks (Trapnell et al. 2012) ou RSEM (Li et Dewey 2011) prennent en compte le fait qu'un exon peut faire partie de plusieurs transcrits et distribuent les reads entre ses différentes formes pour obtenir une quantification plus précise par transcrit.

Avec les comptes bruts de reads par transcrit, les gènes les plus longs sont favorisés : à même niveau d'expression ils auront un nombre de lectures plus élevé. Pour éliminer ce biais, le compte brut est ensuite transformé - le score le plus utilisé pour comparer l'expression des gènes à l'intérieur d'un échantillon est le RPKM (Reads Per Kilobase per Million of mapped reads).

Comparer le transcriptome entre différents échantillons permet d'étudier l'expression des gènes dans différentes conditions expérimentales, au cours du développement ou simplement entre une cellule malade et une cellule saine. La normalisation de la distribution des reads pour autoriser la comparaison de plusieurs échantillons est source de débat et plusieurs méthodes ont été publiées ces dernières années. On peut notamment citer edgeR (Robinson et al. 2010), DESeq2 (Love et al. 2014) et baySeq (Hardcastle et Kelly 2010), trois outils populaires qui utilisent leurs propres statistiques. Il en existe de nombreux autres, et malgré plusieurs études comparant leur performances, aucun consensus n'a été trouvé (Robles et al. 2012, Kvam et al. 2012, Rapaport et al. 2013, Seyednasrollah et al. 2015, Conesa et al. 2016). Toutes les méthodes s'accordent cependant sur l'importance des réplicas dans ces expériences. Certains logiciels ne fonctionneront pas si il n'y en a pas, et il est même communément admis q'un minimum de trois séquençages est nécessaire pour avoir assez de puissance statistique.

4.3. ChIP-Seq

Publiée pour la première fois en 2007 (Barski et al. 2007), le ChIP-Seq est l'adaptation au séquençage à haut débit de la technique de ChIP (Chromatine Immuno-Precipitation). Avec cette technique, un anticorps spécifique d'une protéine d'intérêt interagissant avec l'ADN est utilisé pour identifier les séquences cibles de la protéine par immunoprécipitation de fragments de chromatine. Dans la version originale du protocole de ChIP, les fragments d'ADN immunoprécipités étaient analysés par PCR, ce qui limitait l'étude à un nombre réduit de loci. L'essor des microarrays (ChIP-chip) a permis ensuite d'étendre cette analyse au génome entier (Horak et Snyder, 2002). Le séquençage, comme décrit pour le RNA-Seq, a permis d'améliorer la précision de la localisation des protéines d'intérêt et de quantifier ces interactions (Hu et al. 2015). La technique a été très utilisée pour étudier le positionnement des facteurs de transcription et de variant d'histones (Barski et al. 2007, Johnson et al. 2007, Robertson et al. 2007).

4.3.1. Principe

La première étape du protocole consiste à fixer les protéines sur l'ADN *in vivo*, par l'utilisation de réactifs comme le formaldéhyde ou le DTBP (peroxyde de di-tert-butyle). L'ADN est alors fragmenté en morceaux d'environ 500 bp. Pour l'étude de marques épigénétiques, l'étape de fixation est optionnelle car les histones sont fortement associées à l'ADN. On utilise ensuite un anticorps spécifique de la protéine d'intérêt, fixé à des billes magnétiques, et on sélectionne les portions de l'ADN où la protéine s'est fixée par rétention des billes avec un support magnétique. L'ADN est ensuite libéré des protéines par réversion du crosslink et sert à constituer une librairie pour le séquençage (Fig. 19). L'alignement de ces données sur un génome de référence révèle les positions auxquelles se fixe la protéine.

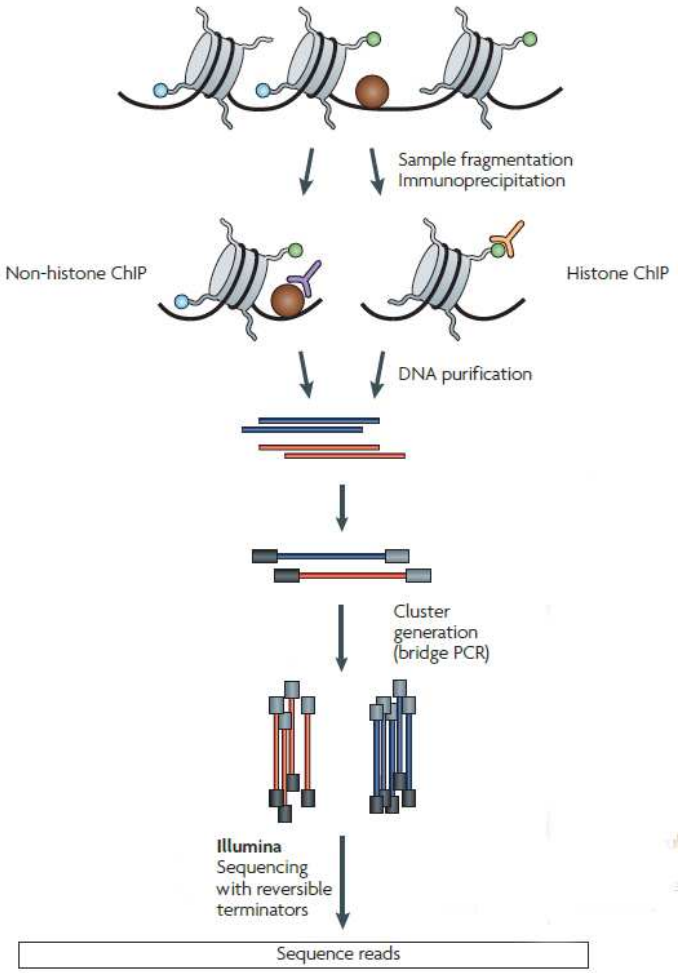


Figure 19: Description d'une expérience de ChIP-Seq. La protéine où la modification d'histone d'intérêt est capturée grâce à un anticorps spécifique en suivant la méthode du ChIP. Les fragments capturés sont purifiés et les adaptateurs, pour le séquençage, sont ajoutés à ses extrémités. L'ADN ainsi préparé peut ensuite être séquencé (Adapté de Park 2009).

4.3.2. Analyse bioinformatique

L'objectif principal de l'analyse des données de ChIP-Seq est d'établir une carte de fixation d'une protéine ou de distribution d'une marque d'histone sur le génome. Il est possible ensuite de faire une comparaison entre plusieurs échantillons du niveau d'accroche des protéines (differential binding). Mais avant ces étapes avancées de l'analyse, les données de ChIP-Seq nécessitent quelques contrôles de qualité supplémentaires.

Contrôle qualité

La nature des données de ChIP-Seq fait que l'on s'attend à voir le signal former des clusters là où la protéine d'intérêt s'est fixée et que cet enrichissement sera séparé sur les brins opposés d'une distance qui dépend de la taille du fragment (Kharchenko et al. 2008).

Une première vérification à effectuer est celle du rapport entre le signal et le bruit de fond. Il existe des mesures pour évaluer la qualité de l'immunoprécipitation, comme la corrélation croisée des reads alignés sur les deux brins (Fig. 20). Le score de corrélation de Pearson est calculé entre les deux brins pour l'enrichissement en reads à chaque position du génome. Un des brins est ensuite décalé d'une position et l'opération est répétée. Le résultat est une courbe des scores de corrélation. En fonction de la taille du décalage, celle-ci produit deux pics dans un ChIP-Seq de bonne qualité. Le premier enrichissement correspond à la taille des reads et est appelé pic fantôme (phantom-peak). Le second pic correspond à la taille du fragment précipité durant l'expérience (Fig. 20). Idéalement le pic fantôme aura une faible valeur et l'enrichissement le plus fort se situe autour de la taille du fragment. Pour évaluer la qualité du ChIP-Seq, deux scores sont calculés à partir de cette courbe. Le NCS (Normalized strand cross-correlation) correspond au rapport entre le score de corrélation pour un décalage de la taille du fragment et le score sans décalage (bruit de fond). Le RSC (relative cross-correlation) correspond au rapport entre les scores pour la taille du fragment et celui pour la taille des reads. Un ChIP-Seq de bonne qualité aura un NCS>1,05 et un RCS>0,8.

La recherche de pics

La recherche de pics (peak-calling) est le point central de l'analyse, il s'agit de définir les positions où la protéine d'intérêt est fixée sur l'ADN. De nouvelles méthodes sont publiées régulièrement et il existe plusieurs publications comparant leur efficacité (Wilbanks et Cacciotti 2010, Kim et al. 2011, Malone et al. 2011, Laajala et al. 2009). Les différentes évaluations ont conclu que le choix du logiciel dépendra de la nature du signal analysé.

Les signaux produits par les ChIP-Seq de facteur de transcription et de marque d'histone diffèrent. Le premier donne un signal très localisé, alors que le second produit un signal étalé (Fig. 21). Il existe donc plusieurs catégories de logiciels. Certains sont spécialisés dans la détection de pics étroits (BayesPeak : Spyrou et al. 2009, CisGenome : Ji et al. 2008), d'autres de zones enrichies (SICER : Xu et al. 2014, RSEG : Song and Smith 2011), d'autres encore essayent de proposer les deux (MACS : Zhang et al. 2008, SPP : Kharchenko et al. 2008).

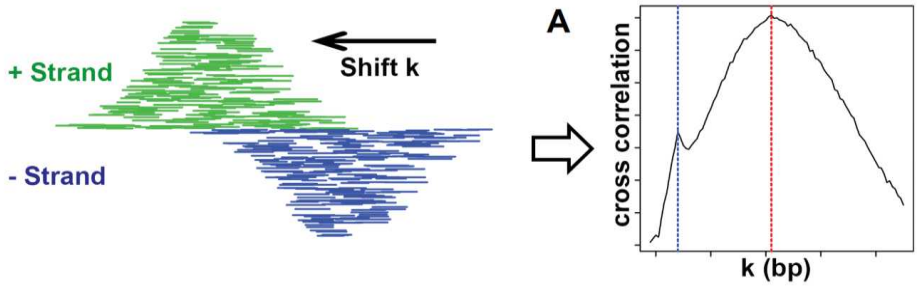


Figure 20 : Illustration de l'analyse de corrélation croisée. L'enrichissement du signal de ChIP-Seq est décalé entre le brin plus et le brin moins (A gauche). (A) En calculant le score de corrélation de l'enrichissement entre les deux brins, pour chaque décalage de K base (Shift k), on obtient une courbe de corrélation croisée. Un premier pic correspond à la taille des read (phantom-peak, indiqué en bleu) et, dans une expérience de bonne qualité, la valeur la plus forte indique la taille du fragment (Indiqué en rouge) (Bailey et al. 2013).

La connaissance du signal étudié, la forme, la taille l'intensité des pics est presque un prérequis pour mener à bien la recherche. Même si les logiciels offrent des valeurs par défaut pour leurs options, celles-ci ne peuvent pas être adaptées à toutes les expériences. Parmi les informations souvent requises, on peut noter la taille du fragment qui sert à décaler le signal des deux brins (Fig. 20, 21), le gap autorisé entre deux enrichissements pour décider par exemple s'il s'agit d'un ou de deux sites de fixation, la taille de la fenêtre utilisée pour scanner le génome - et qui doit être le plus proche possible de la taille du signal recherché -, ou encore le seuil pour les valeurs FDR (False Discovery Rate) ou p-value que ces logiciels accordent le plus souvent aux pics appelé pour mesurer leur qualité.

L'utilisation d'un contrôle est également optionnel - mais fortement recommandé - et certains logiciels ne peuvent pas fonctionner sans. Le contrôle permet d'évaluer l'enrichissement des pics de l'IP et de corriger des variations dues par exemple à des biais d'alignement. Il y a trois types d'échantillons contrôle :

- Input : un séquençage de l'ADN total des cellules étudiées.
- IP aspécifique : le séquençage d'une immunoprécipitation avec un anticorps igG (immunoglobulin G).
- IP dans une autre condition.

L'utilisation d'un input est la solution la plus souvent recommandée sans pour autant qu'il y ait de consensus sur ce point.

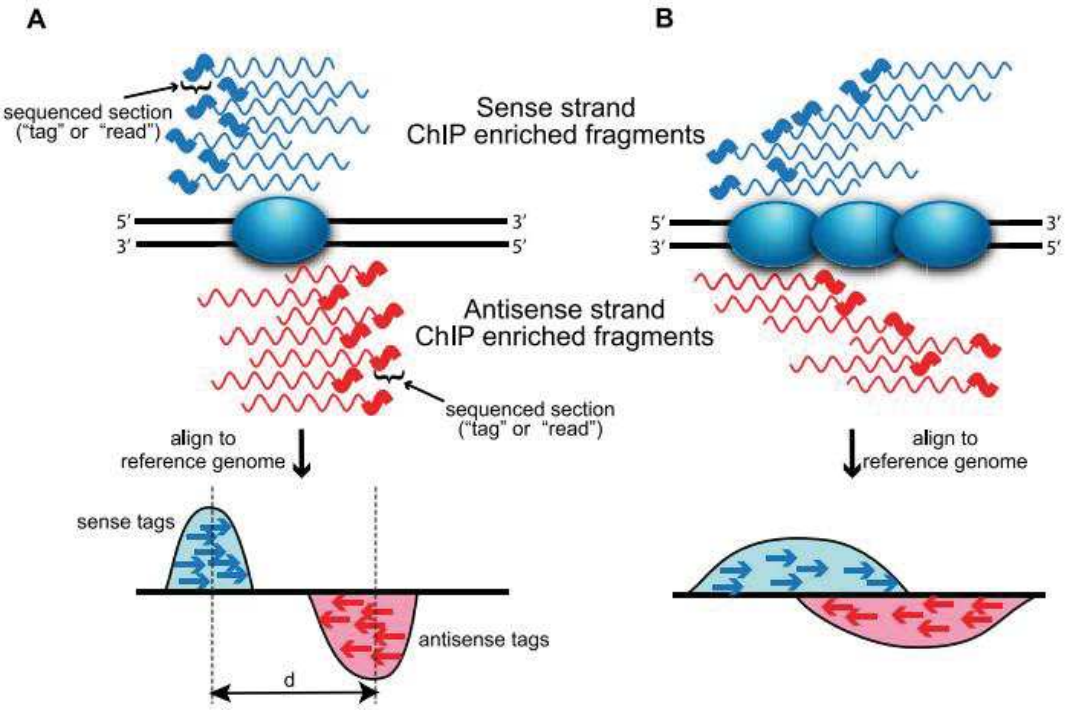


Figure 21: La densité bimodale du signal de ChIP-Seq. (A) Signal étroit, correspondant à une protéine ayant un positionnement dépendant de la séquence de l'ADN (e.g Un facteur de transcription). (B) Signal étalé, dont l'aspect bimodale est plus difficile à distinguer, obtenu pour des événements de fixation distribués dans une zone (e.g Marque d'histone, polymérase) (Wilbanks et Facciotti, 2010).

Evaluation de la reproductibilité de l'IP

Comme pour la plus part des expériences de biologie, il est recommandé de produire au minimum deux immunoprécipitations d'une expérience pour pouvoir tester la reproductibilité du signal et tirer des conclusions solides de l'analyse.

La qualité de deux réplicas peut être testée avant le peak-calling. Le logiciel Deeptools propose de calculer la corrélation de pearson à partir de l'alignement de reads sur le génome. Deux réplicas auront un score d'environ 0,9 alors que deux échantillons sans rapport (i.e IP contre Input) auront un score proche de 0,4.

Après la recherche de pics, on peut calculer le score IDR (Irreproducible Discovery Rate, Li et al 2011). L'analyse requiert un peak-calling qui accorde une p-value ou un score FRD aux pics appelés pour pouvoir les trier. Elle part d'une idée simple, si deux échantillons représentent le même signal, alors les pics réels auront un score de qualité élevé dans les deux échantillons et les faux positifs un score plus variable. A partir du rang d'un pic dans les deux échantillons, le script calcule le score IDR et classe les pics dans deux groupes : reproductibleou non reproductible (Fig. 22). Pour que l'analyse soit efficace, il faut réaliser une recherche de pics souples car pour classer les pics robustes dans le groupe reproductible, l'algorithme a besoin d'informations lui permettant d'évaluer le signal et de bruit de fond.

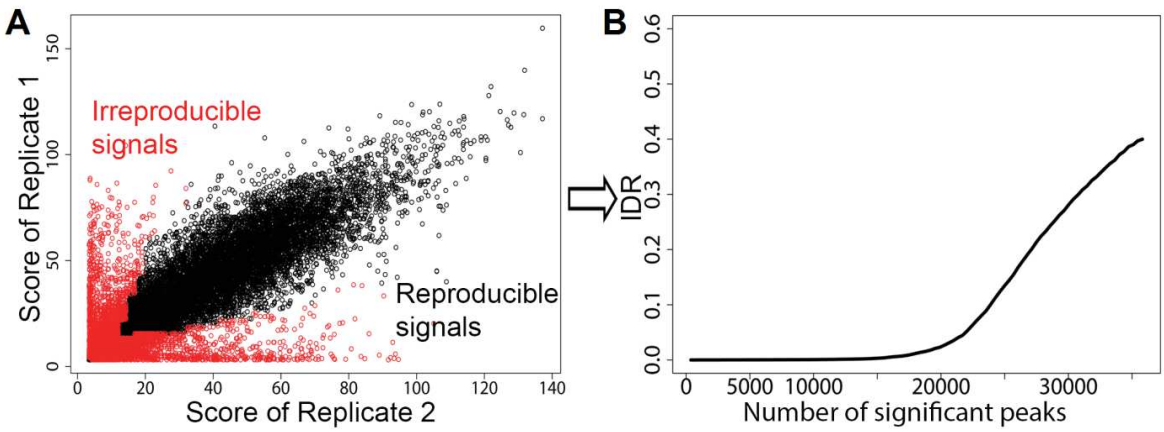


Figure 22: Illustration de l'analyse IDR. (A) Scatterplot des scores par pics dans les deux échantillons. La méthode IDR classe les pics en deux catégories : reproductible (Noir) ou non reproductible (Rouge). (B) La courbe montre le nombre de pics validés par la méthode à différents seuils du score IDR (Bailey et al. 2013).

L'annotation des pics

Une fois un ensemble de positions de fixation de la protéine défini, il est intéressant de savoir si celles-ci sont enrichies pour certaines annotations. Le logiciel CEAS (Shin et al. 2009) propose plusieurs visuels qui permettent d'évaluer l'enrichissement du signal sur certaines annotations, comme le TSS (Transcription Start Site), les exons ou les introns

(annexe 1 un exemple de rapport). Le logiciel propose également de comparer la distribution à celle d'autres signaux. Il suffit pour cela de fournir une liste de positions sur le génome et le programme ajoutera au rapport les valeurs du chevauchement des pics et de l'annotation.

Comparaison de l'intensité de liaison (differential binding)

Pour comparer la fixation d'une même protéine entre deux conditions il est possible de comparer la localisation du signal sur le génome mais aussi l'intensité du signal aux positions enrichies. Plusieurs algorithmes ont été développés pour prendre en compte la variabilité du signal entre deux séquençages, les différences de taille des pics ou encore la quantité de bruit de fond. En 2016, une étude a comparé 14 de ces logiciels (Stainhauser et al. 2016), sur des données réelles et simulées. Les résultats sont pour le moment peu convaincants. Les logiciels ont des performances très variables en fonction du signal étudié et les différentes méthodes ont très peu de chevauchement entre les zones qu'elles appellent. L'étude a même trouvé à plusieurs occasions qu'un simple croisement des données par position donnait de meilleurs résultats que les algorithmes spécialisés.

La recherche de motif

Les facteurs de transcriptions se fixant à des séquences ADN spécifiques, la recherche de motifs récurrents aux sites de fixation trouvés grâce au ChIP-Seq permet d'identifier ses séquences. C'est également utile pour les marques d'histone, car même si leur position ne dépend pas des séquences, elles peuvent être associées au positionnement d'autres protéines.

La recherche de motifs ne se limite pas à la découverte d'une séquence répétée aux sites de fixation. Certains logiciels proposent de comparer les séquences trouvées à d'autres motifs connus ou de chercher s'il existe une séquence voisine qui occupe des positions similaires sur le génome (CentriMo : Bailey et Machanick 2011, SpaMo : Whitington et al. 2011). Il est possible par exemple de découvrir des positions où deux protéines agissent en collaboration ou en compétition.

4.4. DRIP-Seq

Le DRIP-Seq (DNA-RNA ImmunoPrecipitation, Ginno et al. 2012) a été développé pour identifier les positions du génome formant des R-loops *in vivo.* Il fonctionne sur le même principe que le ChIP-Seq, à la différence que l'ADN est déprotéinisé avant son immunoprécipitation avec l'anticorps S9.6, spécifique des hybrides ADN/ARN (Boguslawski et al. 1986). La fragmentation de l'ADN est réalisée avec des enzymes de restriction plutôt

que par sonication pour éviter de déstabiliser les R-loops. De par sa spécificité, l'anticorps ne capture pas les hybrides de très petite taille, comme les amorces des brins d'Okazaki. L'anticorps peut reconnaitre des hybrides de 15 bp in vivo, mais son affinité augmente avec la taille des fragments et le signal est dix fois plus intense pour une séquence supérieure à 30 bp (Dutrow et al. 2008).

Les données produites par le DRIP-Seq sont très similaires à celles du ChIP-Seq et leur analyse peut être menée de la même manière.

4.5. Repli-Seq

Le Repli-Seq (Hansen et al. 2010) a été développé pour caractériser le programme de réplication des cellules. Cette technique a notamment permis d'établir que le timing de réplication variait selon le type cellulaire. Le protocole est adapté d'une technique d'analyse du signal par micro-array. Notre collaborateur Jean-Charles Cadoret a utilisé la méthode par micro-array pour l'analyse du timing de réplication dans nos cellules. J'ai également utilisé les données publiées du Repli-Seq sur les cellules HeLa et c'est la méthode que je vais décrire ici.

4.5.1. Principe

L'ADN en cours de réplication dans une population de cellules asynchrones est marqué pendant 40 minutes – 1h par incorporation de BrdU (5-bromo-2-desoxyuridine). Le BrdU est un analogue de thymidine qui est intégré à l'ADN par les ADN polymérases. Les cellules sont ensuite triées par FACS (Fluorescence-activated cell sorting) en sous-populations correspondant à différences portions de la phase-S (par exemple 6 populations : G1b, S1, S2, S3, S4, G2, Hansen et al. 2010). L'ADN des cellules est ensuite extrait, dénaturé, et l'ADN marqué est immunoprécipité avec un anticorps monoclonal spécifique du BrdU. L'ADN issu de chaque fraction est ensuite utilisé pour fabriquer une librairie et être séquencé (Fig. 23).

4.5.2. Analyse bioinformatique

Les reads obtenus sont positionnés sur le génome en autorisant deux erreurs par alignement. La densité du signal est ensuite calculée pour des fenêtres de 50 kb avec un pas de 1 kb. Le résultat est ensuite normalisé globalement pour avoir un total de 4 millions de lectures par fraction de la phase S. Le pourcentage de signal par fraction de la phase S (G1b, S1, S2, S3, S4, G2) est ensuite calculé à chaque position du génome, ce qui permet de générer

une carte de réplication unique, utilisée pour comparer le timing de réplication entre différents types cellulaires (Fig. 23 B).

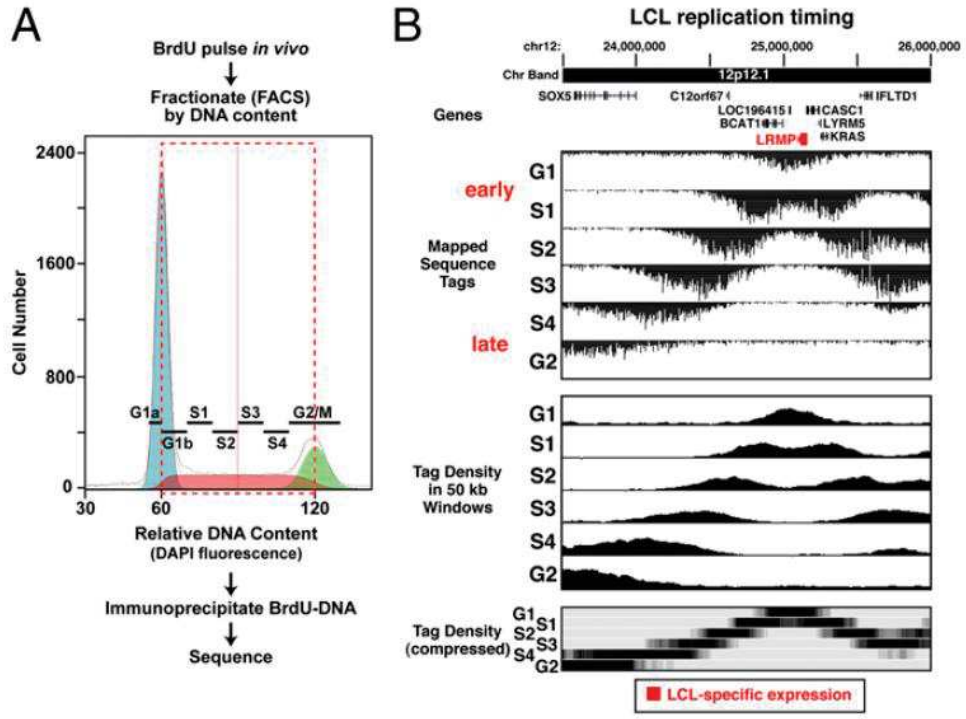


Figure 23 : Description d'une expérience de Repli-Seq. (A) Protocole du Repli-Seq. Après labélisation de l'ADN avec du BrdU, les cellules sont triées par FACS en fonction de leur teneur en ADN, pour former des sous-populations correspondant à différents temps de la phase S. L'ADN marqué par le BrdU est immunoprécipité et utilisé pour constituer une librairie pour le séquençage. (B) Visualisation des résultats sur un locus du chromosome 12. Le signal brut obtenu par alignement est montré en premier (Haut). Ce signal est ensuite compté sur des fenêtres de 50kb et normalisé par la profondeur de séquençage total (milieu). Enfin, l'enrichissement à chaque position est comparé entre les différentes phases. Le profil est normalisé en fonction du pourcentage de signal par phase de réplication (Hansen et al. 2010).

4.6. OK-Seq

Le séquençage des fragments d'Okazaki (OK-Seq, Petryk et al. 2016) permet de déterminer la position des origines de réplication, l'orientation des fourches et les sites de terminaison le long du génome.

4.6.1. Principe

L'ADN naissant est marqué par l'addition d'EdU, un analogue de la thymidine, sur le milieu de croissance d'une population cellules asynchrone. Les fragments d'Okazaki sont

ensuite purifiés par dénaturation de l'ADN et la sélection des séquences inférieures à 200 bp. Les brins marqués sont ensuite sélectionnés grâce à l'ajout d'une biotine par la réaction click et précipités avec des billes strepavidine. L'ADN ainsi récupéré est amplifié et préparé pour un séquençage single-end.

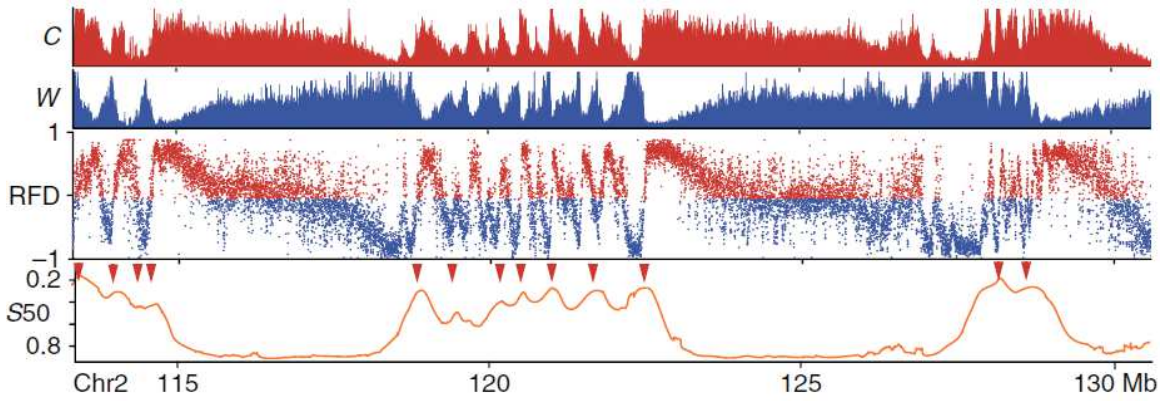


Figure 24 : Profils des données OK-seq. De haut en bas, les fragments d'Okazaki sur le brin Crick (rouge), les fragments sur le brin Watson (bleu), le RFD et le timing de réplication (orange). Les flèches rouges indiquent les positions de pics de réplication précoce (Adapté de Petryk et al. 2015).

4.6.2. Analyse bioinformatique

Le profil représentant l'orientation de la réplication est obtenu en calculant le score RFD (Replication Fork Directionality) dans une fenêtre de 1kb (Fig. 24). Ce score est compris entre -1 et 1, il est obtenu en soustrayant le nombre de lectures alignées sur le brin Crick (C) par le nombre de lectures alignées sur le brin Watson (W) et en divisant le résultat par le nombre total de reads dans la fenêtre.

$$RFD = (C - W) / (C + W)$$

Un score de -1 indique que l'ensemble des fourches sont dirigées vers la gauche, un score de 1 vers la droite et un score de 0, qu'il y a autant de fragments séquencés allant dans les deux sens.

A partir du score RFD et grâce à une analyse HMM (Hidden Markov Model) à quatre états (paramètres Petrik et al. 2016), il est possible de définir les zones d'initiation (AS, Ascending Segment), les zones de terminaison (DS, Descending Segment) et les zones où l'orientation de la réplication est continue dans un sens ou l'autre (FS, Flat Segment).

L'efficacité d'une origine (ΔRFD) est calculée en soustrayant le score RFD dans les 5kb des extrémités de la zone et en divisant le résultat par deux.

$$\Delta RFD = (RFD_{(end)} - RFD_{(start)})/2$$

4.7. GRO-Seq

Le GRO-Seq (Global Run-On Sequencing, Core et al. 2008) a été développé pour capturer la position, la quantité et l'orientation des ARN polymérases engagées dans la transcription à un moment donné (Fig. 25). Ce protocole est complémentaire avec le RNA-Seq. Ce dernier décrit la population d'ARN dans la cellule à un moment donné, mais la stabilité des transcrits est très variable et le RNA-seq ne détecte pas les ARN dont le temps de vie est trop court. La quantité d'ARN ne représente donc pas forcement l'activité de transcription et le GRO-Seq permet de corriger ce problème.

4.7.1. Principe

Comme de nombreuses autres techniques, le GRO-Seq est l'adaptation d'un protocole utilisé avant l'avènement du séquençage de seconde génération. Le principe du NRO (Nuclear Run-On) est de marquer la transcription active grâce à un analogue de l'uridine triphosphate, le BrUTP (5-bromouridine 5'-triphosphate). Pour ne marquer que l'activité des polymérases engagées, l'expérience est menée dans des conditions d'inhibition de la fixation des polymérases. L'ARN est ensuite purifié et fragmenté, puis les brins marqués sont sélectionnés avec un anticorps dirigé contre le BrU. Une librairie est ensuite créée par la transcription inverse de ces ARN afin de générer des ADNc pour le séquençage.

4.7.2. Analyse bioinformatique

Après alignement, les données de GRO-Seq sont comparées à l'annotation de gènes. Il est possible d'analyser visuellement le résultat en faisant un profil moyen aux positions d'intérêt tel que le TSS (Transcription Starting Site) et le TTS (Transcription Termination Site) des gènes.

Pour évaluer l'activité des gènes, le nombre de reads dans le sens de la transcription sur le corps du gène (de 1 kb après le TSS au site de terminaison) est calculé. La probabilité de trouver ce nombre de reads sur un fragment de la taille du gène est calculée avec une distribution de Poison dont la moyenne est le bruit de fond de l'échantillon. Si la probabilité

est inférieure à 0.01 le gène est considéré comme actif (Core et al. 2008). Le début du gène n'est pas utilisé pour ne garder que l'activité d'élongation, le TSS du gène pouvant être occupé par des polymérases en pause.

Pour savoir si le TSS est enrichi en polymérases pausées, l'index de pause est calculé en faisant le ratio de la densité de reads au promoteur sur leur densité sur le corps des gènes. Au-dessus de 2, le score est considéré comme significatif.

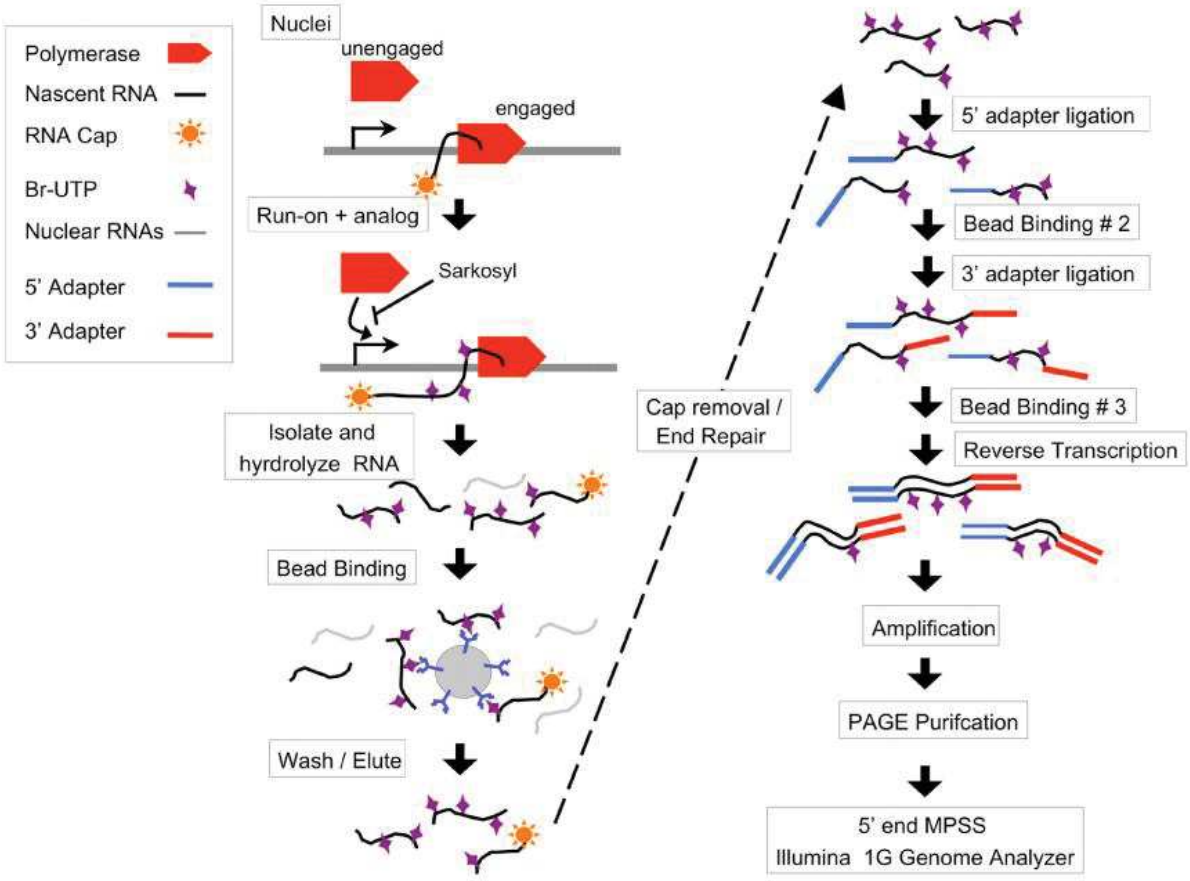


Figure 25 : Description d'une expérience de GRO-Seq. La transcription est marquée par le Br-UTP sur une centaine de bases, en présence de sarkozyle qui empêche l'engagement de nouvelles polymérases. L'ARN est ensuite isolé et hydrolysé pour obtenir des fragments d'une centaine de bases. Les fragments sont capturés avec des billes d'agarose recouvertes d'anticorps spécifique du Br-UTP (α-BrdUTP). Les adaptateurs illumina sont ajoutés aux séquences, d'abord sur la partie 5' puis 3' et la transcription inverse permet de produire des ADNc qui sont finalement amplifiés. La librairie ainsi obtenue est purifiée par la méthode PAGE (Polyacrylamide gel purification) et séquencée (Core et al. 2008).

Objectifs de la thèse

J’ai rejoint l’équipe du Dr Pasero, « Maintien de l'intégrité du génome au cours de la réplication », à l’IGH de Montpellier en décembre 2013. Mon travail consistait à accompagner les biologistes dans la préparation et l’analyse de leurs données de séquençage.

Parmi les projets, il y en avait un qui faisait suite à une étude publiée en 2009, dans laquelle notre équipe avait montré que la déplétion d’ASF/SF2 ou de TOP1 favorise la formation des R-loops. Dans ces cellules, les fourches de réplication progressent moins vite et le signal de γ-H2AX, qui marque les cassures de l’ADN, s’accumule (Tuduri et al. 2009). Ces phénomènes sont corrigés par la surexpression de la RNase H1, qui dégrade la partie ARN des hybrides ARN-ADN, ce qui démontre le lien entre l’accumulation de R-loops et le stress réplicatif. Cependant, à l’époque de ce projet, il n’existait pas encore de protocole pour étudier la formation des R loops à l’échelle du génome. Il était donc impossible de savoir où se formaient les nouveaux hybrides détectés dans les cellules déplétées pour ASF/SF2 et TOP1 et si les R-loops était directement responsables du ralentissement des fourches de réplication.

En 2012, l’équipe du Dr Chédin a mis au point la technique de DRIP-Seq qui permet de visualiser la position des R-loops sur le génome (Ginno et al. 2012). Nous avons donc décidé d’utiliser cette technique ainsi que le ChIP-Seq de γ-H2AX dans nos lignées cellulaires shASF/SF2 et shTOP1 pour étudier l’effet des R-loops sur l’instabilité génomique. Au début de l’année 2014, nous avons obtenu les premiers résultats de séquençage pour ce projet. D’abord le DRIP-Seq qui montrait que les R-loops sont abondantes, y compris dans les cellules contrôles, et se forment principalement sur les gènes. Des résultats qui confirmaient les données publiées par d’autres équipes (Ginno et al. 2013). Puis le ChIP-Seq de γ-H2AX, dont la distribution corrélait avec le signal de DRIP-Seq, mais dont l’étalement du signal ne permettait pas de conclure à une relation directe ou indirecte entre R-loops et cassures de l’ADN.

C’est à ce moment-là, en mai 2014, que j’ai commencé ma thèse. L’objectif de mon projet était de découvrir le rôle des R-loops dans la formation du stress réplicatif. Il fallait pour cela localiser sur le génome les zones favorables à la formation d’hybrides et déterminer les positions où de nouvelles R-loops se forment dans nos lignées déplétées pour ASF/SF2 et TOP1. Nous voulions également caractériser ces positions pour mieux comprendre les mécanismes qui favorisent la formation d’hybrides dans la cellule (annotation, GC%, GC-Skew, état de la chromatine…).

Pour aller plus loin dans l'analyse de la relation entre R-loops et stress réplicatif, nous avons décidé de réaliser plusieurs autres séquençages (RNA-Seq, BLESS, ChIP-Seq de pRPA-S33). Nous voulions confirmer les résultats obtenus avec le γ-H2AX et produire des données qui permettraient de distinguer l'origine du stress de façon plus précise. Nous avons notamment décidé d'utiliser le pRPA-S33 qui marque spécifiquement l'arrêt des fourches de réplication dans la cellule. Le croisement de ces données avec nos résultats de DRIP-Seq devait permettre de mieux comprendre les mécanismes qui entraînent l'instabilité génomique dans les cellules déplétées pour ASF/SF2 et TOP1. Pour cela, en plus des nouvelles données de séquençage que nous allions produire, je devais tirer avantage des informations publiées par d'autres équipes sur notre modèle, les cellules HeLa.

De nombreux rôles ont été décrits pour les R-loops dans la cellule, principalement au niveau de la transcription, où elles favorisent l'initiation et la terminaison, mais aussi dans des processus plus particuliers comme la commutation isotypique des immunoglobulines. Un aspect important du projet était d'étudier la corrélation des différents signaux obtenus dans nos cellules pour comprendre si quelque chose distinguait les hybrides qui jouent un rôle physiologique dans les cellules de ceux qui provoquent le ralentissement de la réplication et l'accumulation de cassure à l'ADN.

Résultats

Il est maintenant bien établi que la mutation d'un seul oncogène (Ras, Myc, CycE,…) est suffisante pour entrainer un stress réplicatif et une augmentation de l'instabilité génétique (Halazonetis et al. 2008). Dans les cellules précancéreuses, l'accumulation de DSBs associée à ce stress oncogénique active p53 et induit l'apoptose ou la sénescence, protégeant ainsi l'organisme contre la progression du processus de tumorigenèse (Vogelstein et al. 2000). Cependant, ce stress chronique exerce une pression continue sur p53, ce qui contribue à sélectionner les événements d'inactivation. Le stress réplicatif induit par les oncogènes pourrait donc expliquer à la fois l'instabilité génomique et la perte précoce de la fonction de p53 dans la plupart des cellules cancéreuses (Halazonetis et al. 2008). Il reste cependant à comprendre le lien existant entre dérégulation des voies oncogéniques et stress réplicatif.

Comme indiqué précédemment, les confits entre réplication et transcription représentent une source potentielle de stress réplicatif. Dans les cellules saines, de nombreux mécanismes permettent de limiter l'impact de ces conflits, notamment par une séparation spatiale et temporelle des activités de réplication et de transcription au cours de la phase S. La dérégulation du métabolisme basal de la cellule ainsi que de l'entrée en phase S perturbe le programme de réplication des cellules et augmente la probabilité d'interférence entre la réplication et la transcription. Le stress réplicatif induit par les oncogènes pourrait donc être causé par la collision des fourches avec les complexes de transcription ou avec des structures produites par la transcription, comme les R-loops. Notre laboratoire a montré précédemment que des cellules humaines et murines déplétées en topoisomérase I (TOP1) et ASF/SF2 présentent un stress réplicatif élevé et une forte instabilité génétique (Tuduri et al. 2009). Les défauts de progression des fourches et l'instabilité génomique étant supprimées dans ces cellules par la surexpression de RNase H1, ces données suggèrent que les R-loops représentent une source majeure de stress réplicatif. Cependant, comme les techniques de cartographies des R-loops n'étaient pas encore disponibles à cette époque, ce lien n'avait pu être directement établi.

Ma thèse a pour objectif de clarifier la nature des relations existant entre la formation de R-loops, le stress réplicatif et l'instabilité génomique. Elle vise également à déterminer si toutes les R-loops sont également toxiques pour la réplication, ou si certaines de ces structures sont particulièrement difficiles à répliquer. Le cas échéant, nous souhaitons déterminer quelles sont les caractéristiques de ces régions à risque du génome. Grâce aux technologies de séquençage à haut débit, nous avons pu établir la répartition sur le génome des R-loops dans nos cellules (DRIP-Seq) et étudier si celles-ci provoquent l'arrêt de la progression des

fourches de réplication (ChIP-Seq de phospho-RPAS33) et l'instabilité génomique (ChIP-Seq de γ-H2AX). Il est important de noter que toutes les études réalisées précédemment sur ces marqueurs de stress réplicatif l'ont été dans des cellules traitées avec de fortes doses d'agents génotoxiques. Etant donné que nous cherchons à détecter un stress spontané dans des cellules en prolifération, nous anticipons des problèmes liés à l'analyse de signaux de bas niveau. Ces travaux ont été réalisés dans des cellules HeLa exprimant un shRNA inductible contre TOP1 ou contre le facteur d'épissage ASF/SF2. Le choix des cellules HeLa comme modèle d'étude est justifié par le fait que de nombreuses données génomiques (ChIP-Seq, GRO-Seq, Repli-Seq, RNA-Seq) sont disponibles dans ces cellules et pourront être incluses dans notre analyse.

1. Le stress réplicatif est causé par les collisions de front avec les R-loops se formant en fin de gènes répliqués précocement

1.1. Introduction

Les conflits entre la réplication et la transcription sont une source majeure d'instabilité génomique. Il a récemment été montré que la progression de la fourche de réplication pouvait être stoppée par des hybrides ARN-ADN, qui se forment au cours de la transcription. Les R-loops sont présentes en abondance sur le génome humain où elles jouent d'importants rôles physiologiques. Il reste à déterminer si toutes les R-loops sont toxiques pour la réplication. Dans le but d'identifier les R-loops qui favorisent l'émergence du stress réplicatif, nous avons cartographié la position, sur le génome humain, des hybrides ARN-ADN et de marqueurs du stress réplicatif (DRIP-Seq, ChiP-Seq). Nous avons créé pour cela des lignées cellulaires HeLa, où la déplétion d'ASF/SF2 ou TOP1 est sous contrôle d'un shRNA inductible. La déplétion de ces deux protéines favorise grandement la formation des R-loops et ralentit la progression des fourches de réplication. Le ChIP-Seq de pRPA-S33 montre qu'une partie seulement des R-loops forment un obstacle aux fourches de réplication, principalement à la fin des gènes activement transcrits et là où la rencontre de la transcription avec la réplication est frontale. Bien que l'effet sur la progression des fourches soit similaire dans les cellules shASF et shTOP1, l'instabilité génomique détectée par γ-H2AX est plus importante dans les cellules déplétées pour TOP1. Ces résultats suggèrent que les R-loops représentent bien un obstacle à la progression des fourches de réplication, mais la présence de stress topologique accentue aussi le stress réplicatif, ce qui pourrait induire les cassures d'ADN et l'instabilité génomique.

La première partie de la publication décrit les caractéristiques des lignées cellulaires mises en place par les biologistes de l'équipe pour pouvoir étudier les R-loops endogènes et leur impact sur le stress réplicatif. La deuxième partie présente les résultats que j'ai obtenus par l'analyse des données de séquençage produites avec ces cellules, ainsi que des données publiées pour les cellules HeLa.

1.2. Publication

(En préparation)

Spontaneous replication stress is caused by head-on collisions with early replicating genes forming R-loops

Alexy Promonet[1]*, Ismaël Padioleau[1]*, Lionel Sanz[2], Anne-Lyne Maurin[1], Magdalena Skrzypczak[3], Amélie Sarrazin[4], Krzysztof Ginalski[3], Frédéric Chedin[2], Malgorzata Rowicka[5], Yea-Lih Lin[1] and Philippe Pasero[1]

1. Institute of Human Genetics, CNRS UMR9002 and University of Montpellier, Equipe labélisée Ligue contre le Cancer, Montpellier France
2. Genome Center, University of California, Davis, California 95616, USA
3. Laboratory of Bioinformatics and Systems Biology, Centre of New Technologies, University of Warsaw, Warsaw, Poland;
4. BioCampus Montpellier, CNRS and University of Montpellier, Montpellier, France
5. Department of Biochemistry and Molecular Biology, UTMB, Galveston, Texas, USA

Corresponding authors:

Philippe Pasero: philippe.pasero@igh.cnrs.fr

Yea-Lih Lin: yea-lih.lin@igh.cnrs.fr

* Equal contribution

ABSTRACT

Co-transcriptional R-loops play important physiological roles in eukaryotic cells, but also induce genomic instability by interfering with DNA replication. Since R-loops cover over 100 Mb of the human genome, we asked whether all these structures are equally toxic to replication forks and if not, what differentiates toxic R-loops from others. To this end, we determined the relative distribution of RNA-DNA hybrids and replication stress signals by DRIP-seq and ChIP-seq in the genome of HeLa cells expressing shRNAs against topoisomerase I and ASF/SF2. This analysis revealed that although most R-loops do not interfere with DNA replication, a specific class of R-loops was frequently associated with phospho-RPA32 (S33), a signal of fork stalling. These R-loops are present at the transcription termination site of highly expressed genes, which are replicated early in S phase and organized in a head-on orientation relative to forks. Remarkably, we also observed an increased γ-H2AX signals in Top1-deficient cells, but not in control and ASF/SF2-deficient cells. Together, these data indicate that spontaneous RS is caused by topological stress occurring upon frontal collisions with a subset of early replicating genes that are prone to form R-loops.

INTRODUCTION

Replication stress (RS) is a major trigger of genomic instability that has been implicated in cancer development. Identifying the regions of the human genome that are intrinsically difficult to replicate is therefore important to understand the etiology of cancer. DNA replication is initiated at thousands of origins distributed throughout the genome, from which replication forks progress bidirectionally (Méchali et al., 2013). Since replication origins are present in a large excess in the human genome and fire with a low frequency, most loci can be replicated by forks progressing in both directions (Petryk et al., 2016). Replication forks stall when they encounter obstacles such as secondary DNA structures, highly transcribed genes or tightly bound protein complexes (Berti and Vindigni, 2016; Zeman and Cimprich, 2014). Prolonged fork arrest may lead to fork collapse and to gross chromosomal rearrangements (Aguilera and Garcia-Muse, 2013).

Stalled replication forks are detected by the intra-S phase checkpoint, a surveillance pathway sensing the presence of excess ssDNA at damaged forks. This checkpoint response is initiated with the binding of the ATR kinase and its partner ATRIP to the ssDNA-binding protein RPA (Zou and Elledge, 2003). Once activated by TopBP1, ATR phosphorylates multiple targets, including the RPA32 subunit of the RPA complex on S33 (called thereafter p-RPA). Fork collapse also leads to the phosphorylation of the histone variant H2AX by ATR (γ-H2AX). Unlike p-RPA, the γ-H2AX signal can spread over several hundreds of kilobases around broken forks (Zeman and Cimprich, 2014). ATR also activates the CHK1 kinase to amplify the checkpoint response, repress late replication origins and prevent premature entry into mitosis (Berti and Vindigni, 2016; Pasero and Vindigni, 2017).

Transcription-replication conflicts (TRCs) represent a major source of replication stress in all organisms, from bacteria to human (Garcia-Muse and Aguilera, 2016; Hamperl and Cimprich, 2016; Merrikh, 2017). This is due to the fact that the replication and transcription machineries share the same DNA template and therefore inevitably interfere with one another. Collisions can occur in a head-on (HO) or codirectional (CD) manner, but frontal collisions are generally considered as more deleterious to genome stability (Garcia-Muse and Aguilera, 2016; Prado and Aguilera, 2005). In bacteria, most genes are organized co-directionally (CD) with forks to limit these collisions (Merrikh, 2017). A similar orientation bias is also present in the human genome, albeit to a lesser extent (Huvet et al., 2007; Petryk et al., 2016).

Replication forks can also encounter three-stranded nucleic acids structures called R-loops. These structures comprise an RNA–DNA hybrid and a displaced DNA strand (Drolet et al., 1995). They are formed during transcription when the nascent RNA reanneals with the template DNA strand, leaving the non-coding strand unpaired (Aguilera and García-Muse, 2012; Chédin, 2016). R-loops are abundant in mammalian genomes and form dynamically over conserved regions such as transcription initiation and termination sites (Ginno et al., 2013; Skourti-Stathaki et al., 2011). They are also involved in multiple physiological processes, such as Cas9-mediated DNA targeting (Jiang and Doudna, 2017), class switch recombination of immunoglobulin genes (Yu et al., 2003), and chromatin patterning (Chédin, 2016). In cells depleted for factors involved in the regulation R-loop metabolism, the pathological accumulation of these structures promotes RS and genomic instability (Garcia-Muse and Aguilera, 2016; Hamperl and Cimprich, 2016). Since RS is relieved by the ectopic expression of RNase H1, a ribonuclease that degrades the RNA strand of RNA-DNA hybrids, it is generally believed that R-loops interfere with the progression of replication forks (Gan et al., 2011; Tuduri et al., 2009; Wellinger et al., 2006). However, the mechanism by which R-loops block replication has remained largely unexplored. Moreover, it is not known whether all R-loops have the potential to act as replication fork barriers in their chromosomal environment, or if specific features determine their toxicity.

We have previously reported that depletion of the DNA topoisomerase 1 (Top1) or the splicing factor ASF/SF2 interferes with replication fork progression and activates the DNA damage response in human and murine cells (Tuduri et al., 2009). Since these defects were suppressed by inhibition of transcription and by the overexpression of RNase H1, we proposed that replication stress was caused by a pathological accumulation of co-transcriptional R-loops. With the development of high-throughput R-loop mapping approaches such as DRIP-seq, it is now possible to investigate the distribution of R-loops in normal and pathological situations in order to characterize the mechanisms that determine their toxicity.

Here, we show that the depletion of Top1 and ASF/SF2 in HeLa cells induces the accumulation of R-loops and RS signals at specific loci. Although RNA-DNA hybrids were detected at thousands of genes in the human genome, only a subset of these genes were enriched in p-RPA32 (S33), a mark of stalled replication forks. This class of genes is characterized with a high level of expression, a very early replication time and a head-on orientation relative to replication forks. Moreover, p-RPA was specifically enriched at the

TTS of these genes, suggesting that R-loops forming at termination zones are more difficult to replicate than those present at promoter regions. Finally, although similar levels of p-RPA were detected at the 3'-end of genes in control, Top1- and ASF/SF2-depleted cells, γ-H2AX was preferentially detected in Top1-depleted cells. Altogether, these data indicate that replication forks pause at the 3'-end of a specific class of R-loop containing may give rise to chromosome breaks, especially when cells are unable to resolve topological constraint occurring at converging replication and transcription complexes.

RESULTS

Depletion of ASF/SF2 and Top1 increases the formation of R-loops in HeLa cells

Depletion of Top1 and ASF/SF2 in human and murine cells induces a chronic replication stress in a transcription-dependent manner. Since RS could be suppressed by the ectopic expression of RNase H1, we proposed that fork stalling was primarily caused by cotranscriptional R-loops (Tuduri et al., 2009). To confirm the implication of R-loops in this process and identify genomic loci involved, we performed a genome-wide analysis of RNA-DNA hybrids in Top1- and ASF/SF2-depleted cells using the S9.6 monoclonal antibody (Boguslawski et al., 1986). To avoid problems associated with the long term toxicity of R-loop structures, we developed original models derived from HeLa cells in which the expression of ASF/SF2 and Top1 can be prevented using tetracycline-inducible shRNAs (Fig. S1A).

To validate our models, we have first quantified the amount of R-loop structures in these cells using an immunofluorescence-based approach combining confocal microscopy and automated image analysis. In HeLa cells depleted for ASF/SF2 (shASF) and Top1 (shTop1), RNA-DNA hybrids were detected as discrete subnuclear foci (green) and as large domains colocalizing with nucleolin (red), corresponding to nucleoli (Fig. 1A). Since this nucleolar signal largely resistant to RNase H digestion (Fig. 1A), we decided to exclude it from the analysis (Fig. S1B). We also excluded cytoplasmic dots corresponding to mitochondria, located outside of nuclei (blue). The number of subnuclear S9.6 foci quantified with CellProfiler showed a two-fold increase in shASF and shTop1 cells relative to control HeLa cells (Fig. 1B). We therefore conclude that the transient depletion of Top1 and ASF/SF2 leads to a significant accumulation of R-loops in HeLa cells.

To monitor the effect of these R-loops on RS in shASF and shTop1 HeLa cells, the progression of individual replication forks was monitored using DNA fiber spreading (Jackson and Pombo, 1998). Cells were labeled for 20 minutes with 5-iodo-2'-deoxyuridine (IdU) and for 20 minute with 5-chloro-2'-deoxyuridine (CldU). DNA fibers were spread on glass slides and incorporation of these halogenated thymidine analogs was detected by immunofluorescence using specific antibodies (Bianco et al., 2012). We measured a 27% reduction in the median length of the CldU tracks in ASF/SF2-depleted cells and a 44% reduction in Top1-depleted cells when compared to control cells (Fig. 1C, S1C), which is consistent with the slow fork phenotype of Top1-deficient human HCT116 cells and murine B

lymphoma-derived P388 cells (Tuduri et al., 2009). This slow fork phenotype was largely suppressed by the transient expression of human RNase H1 in shASF and shTop1 HeLa cells (Fig. 1C), supporting the view that RS is caused by R-loops. Finally, the presence of stress markers such as phospho-RPA32 (S33), phospho-CHK1 (S345) and γ-H2AX in control, shASF and shTop1 HeLa cells was monitored by western blot after cell fractionation. This analysis revealed that depletion of Top1 and to a lesser extent ASF/SF2 induced the activation of CHK1 and the accumulation of γ-H2AX (Fig. 1D), which is consistent with the induction of a chronic RS in these cells.

Genome-wide analysis of the distribution of R-loops by DRIP-seq

To characterize the regions of the human genome that are prone to form R-loops and to induce RS in human cells, we performed a DNA-RNA immunoprecipitation (DRIP) with the S9.6 antibody, followed by next generation sequencing (DRIP-seq) as described (Ginno et al., 2013). Using the MACS2 peak finder, we identified 8726 R-loop peaks in HeLa cells (Fig. 2A, 2B), encompassing ~5% of the human genome, which is consistent with an earlier studies (Ginno et al., 2013; Sanz et al., 2016). The number of peaks identified by MACS2 was higher in shASF (12766) and shTop1 cells (10906) than in control cells. However, the position of the majority of the peaks was conserved among the three cell lines. 80% of these peaks overlapped with coding genes (RefSeq sequences, hg19) and showed a similar distribution relative to gene annotations (Fig. 2C and S2A), with a 3.5-fold enrichment around transcription start sites (TSS) and transcription termination sites (TTS). We also identified 2048 that are specific to shASF and 515 peaks that are specific to shTop1 cells (Fig. 2B). To determine whether differences in gene expression could explain this increased accumulation of R-loops, we performed a RNA-seq analysis in control, shASF and shTop1 cells, but this analysis did not reveal significant differences between cell lines. Together, these data indicate that shASF and shTop1 cells show increased levels of R-loops at specific loci compared to control HeLa cells, which is consistent with the increased number of S9.6 foci detected in these cells by immunofluorescence (Fig. 1B)

In all cell types, the genes with the highest potential to form R-loops showed the highest level of gene expression (Fig. 2D, 2E and 2G). For instance, more than 50% of highly expressed genes (class 10, RPKM > 7.06) were found to form R-loops in HeLa cells and this number raised up to 70% in ASF-depleted cells (Fig. 2D, S2B) and 60% in Top1-depleted cells (Fig. S2B, S2C). Interestingly, the genes forming R-loops specifically in shASF and

shTop1 cells, but not in control cells, were characterized with a moderate expression level (2.8 > RPKM > 0.26, Fig. 2D, Fig. S2C). This suggests that for a given level of expression, the probability to form R-loop increased with the depletion of ASF/SF2 or Top1. Moreover, the depletion of ASF/SF2 and Top1 induced a preferential increase of R-loop levels at TTS relative to TSS (Fig. 2E). This is particularly clear for genes with intermediate expression levels, as shown on heatmaps centered on TTS or TSS (Fig. 2F and S2D). Altogether, these data indicate that the depletion Top1 and ASF/SF2 induces significant changes in the distribution of R-loops in the genome of HeLa cells, with an increased accumulation at TTS and at genes with an intermediate level of expression.

Genes enriched in R-loops accumulate p-RPA and γ-H2AX

Stalled replication forks activate the checkpoint kinase ATR, which detects the presence of excess ssDNA at stalled forks and triggers the phosphorylation of multiple factors, including RPA, H2AX and CHK1 (Zeman and Cimprich, 2014; Zou and Elledge, 2003). To determine whether all the R-loops are equally able to induce RS or whether a specific subset of R-loops is toxic to replication forks, we next analyzed the distribution of phospho-RPA32 (S33) and γ-H2AX by ChIP-seq. As illustrated for shTop1 cells (Fig. 3A), we found that gene-rich regions forming R-loops were globally enriched in p-RPA and γ-H2AX in control, shASF and shTop1 cells. The intensity of p-RPA and γ-H2AX at coding genes was strongly correlated between control cells and shASF or shTop1 (Fig 3B). However, we noticed that the slope of the regression line for the γ-H2AX signal was stronger in shTop1 cells than in shASF cells, suggesting an increased accumulation of γ-H2AX in the absence of Top1. This difference is also visible on heatmaps of genes aligned on TTS (Fig. 3C and S3E). Together, these data indicate that p-RPA accumulates at the same loci and with the same intensity in control and shTop1 or shASF cells, but that shTop1 shows an increased accumulation of γ-H2AX compared to control and shASF cells, which is consistent with western blot analyses (Fig. 1D).

Phospho-RPA is enriched at the TTS of R-loop forming genes

ChIP-seq profiles revealed a sharper distribution of p-RPA signals compared to γ-H2AX. This is consistent with the fact that the distribution of p-RPA is restricted to stalled forks, whereas γ-H2AX signals spread away from stalled or broken forks (Fig. 3C). This is consistent with the broad distribution of γ-H2AX signals around DSBs induced by the AsiSI

restriction enzyme (Iacovoni et al., 2010), used here as positive control for the detection of γ-H2AX-enriched regions (Fig. S3D). To further characterize the regions of the genome that induce replication fork arrest, we next focused on the distribution of p-RPA signals. Interestingly, 90% of the p-RPA peaks identified with MACS2 in shTop1 cells overlapped with genes enriched in R-loops (Fig. 3D). In contrast, only 20% of the genes enriched in R-loops overlapped with p-RPA peaks in the three cell lines (Fig. S3A). Therefore, even though fork arrest occurs primarily at R-loop containing genes, most of these genes do not represent an obstacle for DNA replication.

To discriminate between R-loops that do or do not interfere with fork progression, we compared the distribution of S9.6 and p-RPA signals along a metagene. Although R-loops are enriched at both TSS and TTS, the p-RPA signal was mainly detected at TTS and not at promoter regions, especially in shASF and shTop1 cells (Fig. 3F and S3C). We also observed a 12-fold enrichment of p-RPA peaks at TTS regions relative to the rest of the genome, this enrichment being only of 2.5-fold at TSS (Fig 3E, and S3B). Together, these data suggest that R-loops forming at transcription termination sites are more difficult to replicate than R-loops forming at promoter regions.

Toxic R-loops are found at HO genes replicated early in S

Since only a fraction of R-loop containing genes are enriched in p-RPA, we next searched for specific features that could explain the difference between toxic R-loops and others. Genes are replicated at specific times during S phase, which are determined by a conserved replication timing program. Using available RepliSeq data in HeLa cells (Hansen et al., 2010), we divided genes in six classes (S1 to S6) depending on their time of replication from early to late S. Remarkably, we found that S9.6, p-RPA and γ-H2AX signals were strongly enriched at genes replicating very early in S phase (S1) compared to others (Fig. 4A). Again, this enrichment of R-loops and p-RPA at genes replicated early in S phase was stronger at TTS than at TSS (Fig. S4).

Transcription and Replication conflicts (TRCs) may occur in a head-on (HO) or co-directional (CD) fashion depending on the orientation of genes relative to the direction of fork progression. HO collisions have been shown to be much more detrimental to DNA replication and genome stability when compared to CD collisions (Hamperl and Cimprich, 2016; Prado and Aguilera, 2005). Moreover, two recent reports showed that replication forks approaching highly transcribed genes in a HO orientation - but not in a CD orientation - induce the

formation of R-loops and promote TRCs at engineered loci on the chromosome *Bacillus subtilis* and on an episome in human cells (Hamperl et al., 2017; Lang et al., 2017). To determine whether the orientation of R-loop containing genes affects replication fork arrest in the human genome, we analyzed p-RPA enrichment in regions showing a preferential HO or CD orientation, using published OK-seq data from HeLa cells (Petryk et al., 2016). Global transcription directionality in these regions was derived from GRO-seq analysis in HeLa cells (Andersson et al., 2015). To this end, GRO-seq data were converted into scores for 15 kb windows on each side of strong origins (Fig. 4B). Orientation scores for replication (black arrows) and transcription (blue arrows) were obtained by subtracting left-oriented from right-oriented values, positive values indicate therefore right orientation and negative left orientation. This analysis confirmed that R-loops are preferentially enriched in the HO orientation in comparison to CD (Fig. 4C). Importantly, we also observed that p-RPA signals were enriched at HO genes, compared to CD (Fig. 4C). Together, these data indicate that replication fork stalling is primarily caused by highly transcribed HO genes forming R-loops and replicated early in S phase.

DISCUSSION

A large body of evidence indicates that transcription can interfere with DNA replication, especially when genes are organized in a head-on orientation relative to the direction of fork progression (Garcia-Muse and Aguilera, 2016; Hamperl and Cimprich, 2016; Merrikh, 2017). Since fork arrest can lead to genomic instability, transcription-replication conflicts represent a major threat for the integrity of eukaryotic and prokaryotic genomes. The mechanism by which transcription affects fork progression has been the subject of intense research during the past decades. Besides frontal collisions between replication and transcription complexes, fork arrest can also result from collisions with RNA-DNA hybrids, formed cotranscriptionally when the nascent RNA reanneals with the template DNA behind the RNA polymerase (Chédin, 2016; Hamperl and Cimprich, 2016; Wellinger et al., 2006).

Since the degradation of RNA-DNA hybrids by the overexpression of RNase H1 suppresses replication defects and genomic instability (Gan et al., 2011; Tuduri et al., 2009), it is generally believed that R-loops interfere with DNA replication independently of gene orientation or of the presence of RNA polymerases (Garcia-Muse and Aguilera, 2016). However, this view has been recently challenged by two studies showing that collisions in a head-on orientation increase the formation of R-loops at reporter genes inserted in the *Bacillus subtilis* genome or on an episomal vector maintained in human cells (Hamperl et al., 2017; Lang and Murray, 2011), suggesting that R-loops are not only a cause, but also a consequence of TRC. In both systems, the replication-dependent formation of R-loops was only increased in the HO orientation, indicating that the relative orientation of genes and forks matters. Interestingly, replication fork stalling was conditioned by the ability of the reporter genes to form R-loops, indicating that DNA-RNA hybrids contribute to fork arrest. Finally, a DRIP-seq analysis of the distribution of R-loops in the human genome confirmed that RNA-DNA hybrids more abundant at HO than at CD genes (Hamperl et al., 2017). Together, these data indicate that the formation of R-loops increases when replication forks approach a gene in a HO orientation, presumably because of the accumulation of topological stress ahead of the replication and transcription machineries. Then, these newly formed R-loops would constitute a barrier to fork progression, potentially driving genomic instability. Interestingly, this mechanism is actively used by pathogenic bacteria to increase mutagenesis of virulence and stress genes during infections, in order to promote pathogen adaptation (Lang et al., 2017). Similar mechanisms could also operate in precancerous lesions to promote genomic instability associated with oncogene-induced replication stress. However, the human genome

contains thousands of genes that are prone to form R-loops and that can be replicated in both directions due to the low efficiency of replication origins in vertebrates. Determining whether all the genes prone to form R-loops are equally able to induce replication stress and identifying the key determinants of toxic R-loops represent therefore major challenges.

In this study, we have analyzed the relative distribution of RNA-DNA hybrids and replication stress markers in the genome of HeLa cells expressing shRNAs against Top1 and ASF/SF2, two factors preventing the formation of R-loops. As reported previously (Sanz et al., 2016), we found that R-loops are very abundant in the human genome and form preferentially in the TSS and TTS regions of highly transcribed genes. Our results also indicate that R-loops are more abundant in genes in a HO orientation relative to CD genes, which is consistent with the results discussed above (Hamperl et al., 2017). Interestingly, we also found that only a fraction of R-loop containing genes are enriched in p-RPA, a mark of stalled replication forks, even though 90% of p-RPA enriched regions overlapped with R-loop containing genes. Together, these data indicate that the majority of R-loops can be efficiently replicated, which raises the question of what makes an R-loop toxic for replication forks.

One of the most striking features of R-loop regions enriched in p-RPA is the overlap with transcription termination sites. Indeed, we measured a 12-fold enrichment of p-RPA at the 3'-end of genes. This enrichment was much weaker at TSS (2.5-fold) and gene bodies (1.5-fold), even though these regions contain R-loops. DNA-RNA hybrids forming at termination sites could be more toxic to forks than those forming at promoter regions because of the different nature of R-loop structures. Indeed, unlike R-loops forming at TSS, those enriched at TTS do not show a positive GC skew, (Sanz et al., 2016). These R-loops could be more stable or more resistant to clearance by RNA-DNA helicases or RNase H. Alternatively, replication forks could be more sensitive to R-loops forming at TTS because replication generally initiates within intergenic regions, generating forks that would first encounter TTS in the HO orientation. The fact that gene orientation determines the toxicity of R-loops was also confirmed by focusing on regions of the genome showing a preferential CD and HO orientation, as determined by the analysis of fork direction using OK-seq data (Petryk et al., 2016). In HO regions, we observed an increase of both R-loop and p-RPA signals relative to CD regions. Together, these data suggest that most of the R-loops present in the human genome are not toxic for replication forks. However, a subset of R-loops accumulating at transcription termination sites behave as replication fork barriers when organized in a HO orientation relative to the direction of fork progressing. Finally, we noticed that genes

replicating very early in S phase were more prone to accumulate R-loops and p-RPA. The reason why early replicating genes containing R-loops induce replication stress in the human genome is currently unclear. It is tempting to speculate that these genes could correspond to the early replicating fragile sites (ERFS) mapped by the Nussenzweig group in mouse cells (Barlow et al., 2013). However, it is worth noting that ERFS were mapped in quiescent cells released synchronously into S phase in the presence of replication inhibitors. Here, the mapping of p-RPA and γ-H2AX was performed in untreated cells. Further work is therefore needed to confirm that ERFS correspond to TRCs mediated by R-loops at TTS.

Another surprising observation is that although shASF and shTop1 cells show slow replication forks and similar profiles of R-loop and p-RPA enrichment, γ-H2AX was preferentially detected in shTop1 cells. These data suggest that the slow fork progression and increased fork arrest at R-loop containing genes in shASF cells does not necessarily translate into irreversible fork collapse and extensive phosphorylation of the H2AX, as observed in shTop1 cells. It is therefore likely that shASF cells can deal with transient fork arrests, whereas shTop1 cells failed to restart a fraction of their forks in response to TRCs because of their reduced ability to resolve topological constraints.

In conclusion, our results indicate that although the human genome contains a high density of R-loops, most of these structures are not harmful to DNA replication. Yet, we identified a subset of R-loop containing genes that are replicated very early in S phase and that are enriched in p-RPA, a mark of stalled replication forks. This mark was particularly enriched at TTS, presumably because transcription termination generates structures that are intrinsically difficult to replicate due to the accumulation of R-loops and/or stalled RNA polymerases. This effect was particularly clear for genes organized in a head-on manner relative to the direction of fork progression, indicating that orientation is a key determinant of RS induced by R-loops. Finally, we observed a specific enrichment of γ-H2AX in Top1-depleted cells relative to control or ASF/SF2-depleted cells. These data indicate that topological stress contributes to TRCs by converting transiently arrested forks into chromosomal breaks.

REFERENCES

Aguilera, A., and Garcia-Muse, T. (2013). Causes of genome instability. Annu Rev Genet *47*, 1-32.

Aguilera, A., and García-Muse, T. (2012). R Loops: From Transcription Byproducts to Threats to Genome Stability. Molecular Cell *46*, 115-124.

Andersson, R., Chen, Y., Core, L., Lis, J., Sandelin, A., and Jensen, T.H. (2015). Human gene promoters are intrinsically bidirectional. Molecular Cell *60*, 346-347.

Barlow, J.H., Faryabi, Robert B., Callén, E., Wong, N., Malhowski, A., Chen, Hua T., Gutierrez-Cruz, G., Sun, H.-W., McKinnon, P., Wright, G., *et al.* (2013). Identification of Early Replicating Fragile Sites that Contribute to Genome Instability. Cell *152*, 620-632.

Berti, M., and Vindigni, A. (2016). Replication stress: getting back on track. Nat Struct Mol Biol *23*, 103-109.

Bianco, J.N., Poli, J., Saksouk, J., Bacal, J., Silva, M.J., Yoshida, K., Lin, Y.-L., Tourrière, H., Lengronne, A., and Pasero, P. (2012). Analysis of DNA replication profiles in budding yeast and mammalian cells using DNA combing. Methods *57*, 149-157.

Boguslawski, S.J., Smith, D.E., Michalak, M.A., Mickelson, K.E., Yehle, C.O., Patterson, W.L., and Carrico, R.J. (1986). Characterization of monoclonal antibody to DNA.RNA and its application to immunodetection of hybrids. Journal of immunological methods *89*, 123-130.

Breslin, C., Clements, P.M., El-Khamisy, S.F., Petermann, E., Iles, N., and Caldecott, K.W. (2006). Measurement of chromosomal DNA single-strand breaks and replication fork progression rates. Methods Enzymol *409*, 410-425.

Chédin, F. (2016). Nascent Connections: R-Loops and Chromatin Patterning. Trends in Genetics *32*, 828-838.

Drolet, M., Phoenix, P., Menzel, R., Masse, E., Liu, L.F., and Crouch, R.J. (1995). Overexpression of RNase H partially complements the growth defect of an Escherichia coli delta topA mutant: R-loop formation is a major problem in the absence of DNA topoisomerase I. Proc Natl Acad Sci U S A *92*, 3526-3530.

Gan, W., Guan, Z., Liu, J., Gui, T., Shen, K., Manley, J.L., and Li, X. (2011). R-loop-mediated genomic instability is caused by impairment of replication fork progression. Genes & Development *25*, 2041-2056.

Garcia-Muse, T., and Aguilera, A. (2016). Transcription-replication conflicts: how they occur and how they are resolved. Nat Rev Mol Cell Biol *17*, 553-563.

Ginno, P.A., Lim, Y.W., Lott, P.L., Korf, I., and Chédin, F. (2013). GC skew at the 5′ and 3′ ends of human genes links R-loop formation to epigenetic regulation and transcription termination. Genome Research *23*, 1590-1600.

Hamperl, S., Bocek, M.J., Saldivar, J.C., Swigut, T., and Cimprich, K.A. (2017). Transcription-Replication Conflict Orientation Modulates R-Loop Levels and Activates Distinct DNA Damage Responses. Cell *170*, 774-786.e719.

Hamperl, S., and Cimprich, K.A. (2016). Conflict Resolution in the Genome: How Transcription and Replication Make It Work. Cell *167*, 1455-1467.

Hansen, R.S., Thomas, S., Sandstrom, R., Canfield, T.K., Thurman, R.E., Weaver, M., Dorschner, M.O., Gartler, S.M., and Stamatoyannopoulos, J.A. (2010). Sequencing newly replicated DNA reveals widespread plasticity in human replication timing. Proceedings of the National Academy of Sciences *107*, 139-144.

Huvet, M., Nicolay, S., Touchon, M., Audit, B., d'Aubenton-Carafa, Y., Arneodo, A., and Thermes, C. (2007). Human gene organization driven by the coordination of replication and transcription. Genome Res *17*, 1278-1285.

Iacovoni, J.S., Caron, P., Lassadi, I., Nicolas, E., Massip, L., Trouche, D., and Legube, G. (2010). High-resolution profiling of [gamma]H2AX around DNA double strand breaks in the mammalian genome. EMBO J *29*, 1446-1457.

Jackson, D.A., and Pombo, A. (1998). Replicon clusters are stable units of chromosome structure: evidence that nuclear organization contributes to the efficient activation and propagation of S phase in human cells. J Cell Biol *140*, 1285-1295.

Jiang, F., and Doudna, J.A. (2017). CRISPR–Cas9 Structures and Mechanisms. Annual Review of Biophysics *46*, 505-529.

Kamentsky, L., Jones, T.R., Fraser, A., Bray, M.-A., Logan, D.J., Madden, K.L., Ljosa, V., Rueden, C., Eliceiri, K.W., and Carpenter, A.E. (2011). Improved structure, function and compatibility for CellProfiler: modular high-throughput image analysis software. Bioinformatics *27*, 1179-1180.

Lang, G.I., and Murray, A.W. (2011). Mutation rates across budding yeast chromosome VI are correlated with replication timing. Genome Biol Evol *3*, 799-811.

Lang, K.S., Hall, A.N., Merrikh, C.N., Ragheb, M., Tabakh, H., Pollock, A.J., Woodward, J.J., Dreifus, J.E., and Merrikh, H. (2017). Replication-Transcription Conflicts Generate R-Loops that Orchestrate Bacterial Stress Survival and Pathogenesis. Cell *170*, 787-799.e718.

Lin, Y.L., Noel, D., Mettling, C., Reant, B., Clot, J., Jorgensen, C., and Corbeau, P. (2004). Feline immunodeficiency virus vectors for efficient transduction of primary human synoviocytes: application to an original model of rheumatoid arthritis. Hum Gene Ther *15*, 588-596.

Méchali, M., Yoshida, K., Coulombe, P., and Pasero, P. (2013). Genetic and epigenetic determinants of DNA replication origins, position and activation. Current Opinion in Genetics & Development *23*, 124-131.

Merrikh, H. (2017). Spatial and Temporal Control of Evolution through Replication–Transcription Conflicts. Trends in Microbiology *25*, 515-521.

Pasero, P., and Vindigni, A. (2017). Nucleases acting at stalled forks: how to reboot the replication program with a few shortcuts. Annual Review in Genetics *in press*.

Petryk, N., Kahli, M., d'Aubenton-Carafa, Y., Jaszczyszyn, Y., Shen, Y., Silvain, M., Thermes, C., Chen, C.-L., and Hyrien, O. (2016). Replication landscape of the human genome. Nat Commun *7*.

Prado, F., and Aguilera, A. (2005). Impairment of replication fork progression mediates RNA polII transcription-associated recombination. EMBO J *24*, 1267-1276.

Sanz, Lionel A., Hartono, Stella R., Lim, Yoong W., Steyaert, S., Rajpurkar, A., Ginno, Paul A., Xu, X., and Chédin, F. (2016). Prevalent, Dynamic, and Conserved R-Loop Structures Associate with Specific Epigenomic Signatures in Mammals. Molecular Cell.

Skourti-Stathaki, K., Proudfoot, Nicholas J., and Gromak, N. (2011). Human Senataxin Resolves RNA/DNA Hybrids Formed at Transcriptional Pause Sites to Promote Xrn2-Dependent Termination. Molecular Cell *42*, 794-805.

Tuduri, S., Crabbé, L., Conti, C., Tourrière, H., Holtgreve-Grez, H., Jauch, A., Pantesco, V., De Vos, J., Thomas, A., Theillet, C.*, et al.* (2009). Topoisomerase I suppresses genomic instability by preventing interference between replication and transcription. Nature Cell Biology *11*, 1315-1324.

Wellinger, R.E., Prado, F., and Aguilera, A. (2006). Replication Fork Progression Is Impaired by Transcription in Hyperrecombinant Yeast Cells Lacking a Functional THO Complex. Mol Cell Biol *26*, 3327-3334.

Yu, K., Chedin, F., Hsieh, C.-L., Wilson, T.E., and Lieber, M.R. (2003). R-loops at immunoglobulin class switch regions in the chromosomes of stimulated B cells. Nat Immunol *4*, 442-451.

Zeman, M.K., and Cimprich, K.A. (2014). Causes and consequences of replication stress. Nat Cell Biol *16*, 2-9.

Zou, L., and Elledge, S.J. (2003). Sensing DNA Damage Through ATRIP Recognition of RPA-ssDNA Complexes. Science *300*, 1542-1548.

STAR METHODS

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Antibodies		
Mouse anti-BrdU clone B44	BD Biosciences	Cat #347580
Rat anti-BrdU clone BU1/75	Eurobio Abcys	Cat #ABC117-7513
Mouse anti-ssDNA	Millipore	Cat #MAB3868
Rabbit anti-pCHK1 (S345)	Cell Signaling Technology	Cat #2348
Mouse anti-γ-H2AX (S139) for WB	Millipore	Cat #05-636
Rabbit anti-γ-H2AX (S139) for ChIP	ABGENT	Cat #AJ1351a
Mouse anti-RNA-DNA hybrid S9.6 hybridoma	ATCC	Cat #HB8730

Rabbit anti-pRPA (S33)	Bethyl	Cat #A300246A
Rabbit anti-Nucleolin	Abcam	Cat #ab22758
Rabbit anti Rnase H1	Santa Cruz	Cat #sc-30319
Rabbit anti-Actin	Sigma-Aldrich	Cat #A4700
Rat anti-Tubulin	Abcam	Cat #ab6161
Rabbit anti-TBP	Cell Signaling Technology	Cat #4302
Rabbit anti- Histone H3	Abcam	Cat #ab1791
Mouse anti-ASF/SF2	Molecular Probes Invitrogen	Cat #32-4500
Rabbit anti-TOP1	Abcam	Cat #ab3825
Chemicals, Peptide, and Recombinant Proteins		
5-iodo-2'-deoxyuridine (IdU)	MP Biomedicals	Cat #2100357
5-chloro-2'-deoxyuridin (CldU)	MP Biomedicals	Cat #2105478
Poly-D-Lysine	Sigma-Aldrich	Cat #P4707
(Hexadimethrine bromide (Polybrene)	Sigma-Aldrich	Cat #H9268
Deoxycycline	Sigma-Aldrich	Cat #D9891
Agarose protein A/G beads	Pierce	Cat #20241
Sepharose protein A beads	Sigma-Aldrich	Cat #P2545
Sepharose protein G beads	Sigma-Aldrich	Cat #P3296
Critical Commercial Assays		
Ipure	Diagenode	Cat#C03010012
TruSeq ChIP Library Preparation Kit	Illumina	Cat#IP-202-1024
ThruPLEX® DNA-seq Kit	Rubicon Genomics	R400407
jetPEI DNA Transfection Reagent	Polyplus-transfection	101-10N

Experimental Models: Cell Lines		
Human: HEK293T	ATCC	CRL-3216
Human: HeLa	ATCC	CCL2
Recombiant DNA		
pLVX-Tet-on	CLONTECH Laboratories	Cat#632162
TRIPZ-shASF/SF2	Dharmacon	RHS4696-99711654
TRIPZ-shTOP1	Dharmacon	RHS4696-99707195
RNase H1-GFP	Pommier laboratory	N/A
pEGFP-N1	CLONTECH Laboratories	Cat#6085-1
Software and Algorithms		
ImageJ	ImageJ	https://imagej.nih.gov/ij/
CellProfiler cell image analysis software	Carpenter Lab.	http://cellprofiler.org/
MetaMorph	Molecular Device	https://www.moleculardevices.com/systems/metamorph-research-imaging/metamorph-microscopy-automation-and-image-analysis-software
Oligonucleotides		
MYADM Forward (RNA-DNA positive): CGTAGGTGCCCTAGTTGGGAG	Ginno et al. 2012	N/A
MYADM Reverse (RNA-DNA positive) : TCCATTCTCATTCCCAAACC	Ginno et al. 2012	N/A

RPL13A Forward (RNA-DNA positive): AATGTGGCATTTCCTTCTCG	Ginno et al. 2012	N/A
RPL13A Reverse (RNA-DNA positive): CCAATTCGGCCAAGACTCTA	Ginno et al. 2012	N/A
SNRPN Forward (RNA-DNA negative): GCCAAATGAGTGAGGATGGT	Ginno et al. 2012	N/A
SNRPN Reverse (RNA-DNA negative): TCCTCTCTGCCTGACTCCAT	Ginno et al. 2012	N/A
EGR1 downstream Forward (RNA-DNA negative): GAACGTTCAGCCTCGTTCTC	Ginno et al. 2012	N/A
EGR1 downstream Reverse (RNA-DNA negative): GGAAGGTGGAAGGAAACACA	Ginno et al. 2012	N/A
Chr 22 Forward (γ-H2AX negative): CCCATCTCAACCTCCACACT	Iacovoni et al. 2010	N/A
Chr22 Reverse (γ-H2AX negative) : CTTGTCCAGATTCGCTGTGA	Iacovoni et al. 2010	N/A
Chr1 Forward (γ-H2AX positive) : TTCCTGCAGCCTCATTTTCT	Iacovoni et al. 2010	N/A
Chr1 Reverse (γ-H2AX positive): TGATGATGCCTTTTCCCTTC	Iacovoni et al. 2010	N/A

EXPERIMENTAL MODEL AND SUBJECT DETAILS

Cell culture

Human embryonic kidney (HEK) HEK293T cells (ATCC CRL-3216) and HeLa cells were cultured in Dulbecco's modified Eagle's medium (DMEM) supplemented with 10% fetal calf serum (FCS) at 37°C in 5% CO_2.

METHODS DETAILS

Production of lentiviral vectors and cell transduction

HIV-1-derived lentiviral vectors were produced in HEK293T cells as previously described (Lin et al., 2004). Cells were seeded on poly-D-lysine coated plates and transfected with packaging

plasmid (psPAX2, Addgene plasmid #12260): transfer vector (pLVX-Tet-on; TRIPZ-shTOP1; TRIPZ-shASF/SF2): vesicular stomatitis virus envelop plasmid (pMD2.G, plasmid #12259) at a ratio 5:3:2 by the calcium phosphate method. The culture medium was collected 48h post-transfection, filtrated using 0.45 µm filters and concentrated at 100 folds by ultracentrifugation at 89,000 xg at 4 °C for 1h30. HeLa cells were transduced at a M.O.I= 10 (Multiplicity of Infection) by centrifugation at 1500 xg at 30°C for 1h30 in the presence of 5 µg/ml of polybrene.

DNA fiber spreading

DNA fiber spreading was performed as described previously (Breslin et al., 2006; Jackson and Pombo, 1998). Briefly, subconfluent cells were sequentially labeled first with 10 µM 5-iodo-2'-deoxyuridine (IdU) and then with 100 µM 5-chloro-2'-deoxyuridine (CldU) for the indicated times. One thousand cells were loaded onto a glass slide (StarFrost) and lysed with spreading buffer (200 mM Tris-HCl pH 7.5, 50 mM EDTA, 0.5% SDS) by gently stirring with a pipette tip. The slides were tilted slightly and the surface tension of the drops was disrupted with a pipette tip. The drops were allowed to run down the slides slowly, then air dried, fixed in methanol/acetic acid 3:1 for 10 minutes, and allowed to dry. Glass slides were processed for immunostaining with mouse anti-BrdU to detect IdU, rat anti-BrdU to detect CldU, mouse anti-ssDNA antibodies (see Supplemental Information for details) and corresponding secondary antibodies conjugated to various Alexa Fluor dyes. Nascent DNA fibers were visualized by using immunofluorescence microscopy (Leica DM6000 or Zeiss ApoTome). The acquired DNA fiber images were analyzed by using MetaMorph Microscopy Automation and Image Analysis Software (Molecular Devices) and statistical analysis was performed with GraphPad Prism (GraphPad Software). The lengths of at least 150 CldU tracks were measured per sample.

Detection of RNA-DNA hybrids by immunofluorescence confocal microscopy

Cells growing on coverslips were fixed in pre-chilled 100% methanol at -20°C for 5 minutes and washed with 1x PBS for three times. The coverslips were incubated with the anti-RNA-DNA hybrid antibody (S9.6 at 1 mg/ml) and anti-nucleolin antibody at the dilution of 1:300 and 1:1000 respectively overnight at 4°C after blocking in PBS containing 1% BSA in PBST (1X PBS/0.1 % Tween 20) for 1h at room temperature. The RNA-DNA hybrids were detected through the incubation with a secondary antibody conjugated to Alexa Fluor dye, followed by DAPI staining and mounted with ProlongGold (Invitrogen). For RNaseH treatment, the coverslips were incubated with 5 units of recombinant RNase H (New England BioLabs) in 50 µl of blocking buffer overnight at 37°C before incubating with first antibodies. Images were acquired by using a Zeiss LSM780 confocal microscope. Numbers of nucleoli-excluded nuclear RNA-DNA hybrid foci were quantified by using the CellProfiler cell image analysis software (Kamentsky et al., 2011).

Detection of RNA-DNA hybrids by slot blotting

Cells were lysed in 0.5% SDS/TE, pH8.0 containing Proteinase K overnight at 37°C. Total DNA was isolated with Phenol/Chloroform/Isoamylalcohol extraction followed by standard ethanol precipitation and quantified using Nanodrop. Half microgram of total DNA was loaded in duplicate onto a Hybond-N^+ membrane using slot blot apparatus. The membrane was separated in two, one for direct UV crosslinking at 0.12 Joules and the other for DNA denaturation. To denature DNA, membrane was incubated with denaturation buffer (0.5 M NaOH; 1.5 M NaCl) for 10 minuTTS and neutralization buffer (1 M NaCl and 0.5 M Tris, pH7.5) for another 10 minuTTS prior to UV crosslinking. Membranes were blocked with 5% skim milk in PBST (PBS; 0.1% Tween-20) for 1hr. The RNA-DNA hybrids and ssDNA were detected by immunoblotting.

Chromatin fractionation

Cells were incubated with CSK-Triton lysis buffer (10 mM PIPES, pH6.8; 100 mM NaCl; 1 mM $MgCl_2$; 1 mM EGTA; 300 mM Sucrose; 10 mM DTT; 0.2% Triton X-100; protease inhibitor; phosphatatse inhibitor) on ice for 10 minutes, and harvested by scraping. The supernatant was collected after centrifugation at 3000 rpm for 5 minutes at 4°C. Pellet was resuspended in CSK-Triton buffer and incubated for 10 minutes on ice. Another round of centrifugation at 3000 rpm for 5 minuTTS at 4°C was performed to separate nucleoplasm and chromatin fractions, supernatant and pellet, respectively.

Chromatin immunoprecipitation sequencing (ChIP-seq)

ChIP assays were carried out according to the protocol described in (Tyteca et al. 2006). Formaldehyde was added to the culture medium to a final concentration of 1% for10 minutes at room temperature. Glycine was added to a final concentration of 0.125 M for 5 minutes to stop crosslinking. Cells were harvested by scraping after PBS wash. Pelleted cells were lysed in lysis buffer (5mM PIPES, pH 8; 85 mM KCl; 0.5% NP-40). The lysates were homogenized with a Dounce homogeniser and nuclei were harvested by centrifugation. Nuclei were then incubated in nuclear lysis buffer (50mM Tris, pH 8.1; 10mM EDTA; 1% SDS) and sonicated at 70 % amplitude for a duration of 3min25sec with 15" ON and 45" OFF (sonicator Qsonica Q700) to obtain DNA fragments of about 500–1000 bp. Samples were diluted 10 times in dilution buffer (0.0 % SDS; 1. % Triton X-100; 1.2 mM EDTA; 16.7 mM Tris, pH 8.1; 167 mM NaCl) and subjected to a 45 min preclearing with 140 µl of previously blocked protein-A and protein-G beads. Blocking was achieved by incubating the agarose beads with 500 µg of BSA and 200 µg of herring sperm DNA for 3 h at 4°C. Precleared samples were incubated overnight at 4°C with antibodies specific for γ-H2AX (10 µl) or without antibody as negative control. Immune complexes were then recovered by incubating the samples with140 µl of blocked protein A/protein G beads for 2 hours at 4°C on a rotating wheel. Beads were washed once in dialysis buffer (2 mM EDTA; 50 mM Tris, pH 8; 0.2% Sarkosyl) and four times in wash buffer (100 mM Tris, pH 8.8; 500 mM LiCl; 1% NP-40; 1% NaDoc). Elution from the beads was achieved by incubation in elution buffer (1% SDS; 100 mM $NaHCO_3$) for 15 minuTTS. Crosslink was reversed by adding NaCl and RNase A to the samples and incubating overnight at 62°C. After a 2 hr-proteinase K treatment, DNA was precipitation by Phenol/Chloroform extraction and ethanol precipitation. The A*si*SI-ER-U2OS cells treated with or without hydroxytamoxifen (4-OHT) were included as positive control for the validation of experiments (Caron et al. 2012). The pulled down material and input DNA were then size-selected, and ligated to Illumina barcoded adaptors, using TruSeq ChIP Sample Preparation Kit (Illumina) or ThruPLEX® DNA-seq Kit (Rubicon Genomics) for sequencing on Illumina HiSeq2500 and HiSeq4000 platforms.

For phosphoRPA-S33 ChIP, similar procedure was performed with minor modifications. Cells were resuspended in sonication buffer (50 mM HEPES, pH8.0; 140 mM NaCl; 1 mM EDTA; 1% Triton X-100; 0.1% NaDoc; 0.5% SDS) and proceeded to sonication. Immunoprecipitation was performed using 30 µg chromatin and 4 µg oanti-phosphoRPA-S33

antibody. The pulldown material was eluted using IPure kit (Diagnode) and proceeded to NGS as described above.

DNA/RNA immunoprecipitation sequencing (DRIP-seq)

DRIP-Seq was performed as described previously (Ginno et al. 2012). Cells (5 x 10^6) were lysed in 0.5% SDS/TE, pH8.0 containing Proteinase K overnight at 37°C. Total DNA was isolated with Phenol/Chloroform/Isoamylalcohol extraction followed by standard ethanol precipitation. One-third of total DNA was fragmented by a cocktail of restriction enzymes (EcoRI, HindIII, BsrgI, SspI, XbaI) overnight at 37°C. A negative control treated overnight with RNase H was included. Digested DNA was purified by Phenol/Chloroform/Isoamylalcohol extraction, ethanol precipitation and quantified by Nanodrop. Four micrograms of digested DNA was diluted in binding buffer (10 mM $NaPO_4$, pH7.0; 0.14 M NaCl; 0.05% Triton X-100) and incubated with 10 µg of S9.6 antibody overnight at 4 °C on a rotator. DNA/antibody complexes were added for 2 hr at 4°C to Agarose Protein A/G beads prewashed with binding buffer. Immunoprecipitated DNA was eluted by incubating with elution buffer (50 mM Tris pH8.0; 10 mM EDTA; 0.5% SDS) containing Proteinase K at 55°C for 45 minutes on a rotator. The eluent was precipitated by Phenol/Chloroform/Isoamylalcohol extraction and ethanol precipitation. Validation of DRIP procedure was performed by qPCR (see KEY RESOURCES TABLE for primer sequences). The pulled down material and input DNA were then sonicated, size-selected, and ligated to Illumina barcoded adaptors, using TruSeq ChIP Sample Preparation Kit (Illumina) for sequencing on Illumina HiSeq2500 and HiSeq4000 platforms.

Bioinformatic analyses

The quality of sequencing data was assessed with FastQC and homemade PERL and python scripts (FastQC :http://www.bioinformatics.babraham.ac.uk/projects/fastqc). ChIP-seq and DRIP-seq data were aligned to Human genome reference (hg19 assembly) with Bowtie2 (Langmead et al., 2012) and RNA-seq using STAR (Dobin et al., 2013). Mapping quality was assessed with SAMtools (Li et al., 2009) and homemade python scripts. Peak-calling for DRIP-seq data was done using MACS2 (Zhang et al., 2008) with a q-value of 0.05 and keeping up to five duplicates. Reproducible peaks from replicates where then selected using the Irreproducible Discovery Rate (IDR) method from ENCODE Project (Landt *et al.* 2012), with a cutoff value of 0.05. Intersection of transcripts annotation (RefSeq hg19) with R-loop signal was done using BEDTools (Quinlan and Hall, 2010). DeepTools2 (Ramírez et al.,

2012) was used to compute and draw, enrichment heatmaps and profiles on positions of interest (peaks, TSS, TTS). Further analyses were done in R (), with Bioconductor packages (Gentleman et al., 2004) and ggplot2 (Wickham, 2009) for graphic representation.

FIGURE 1

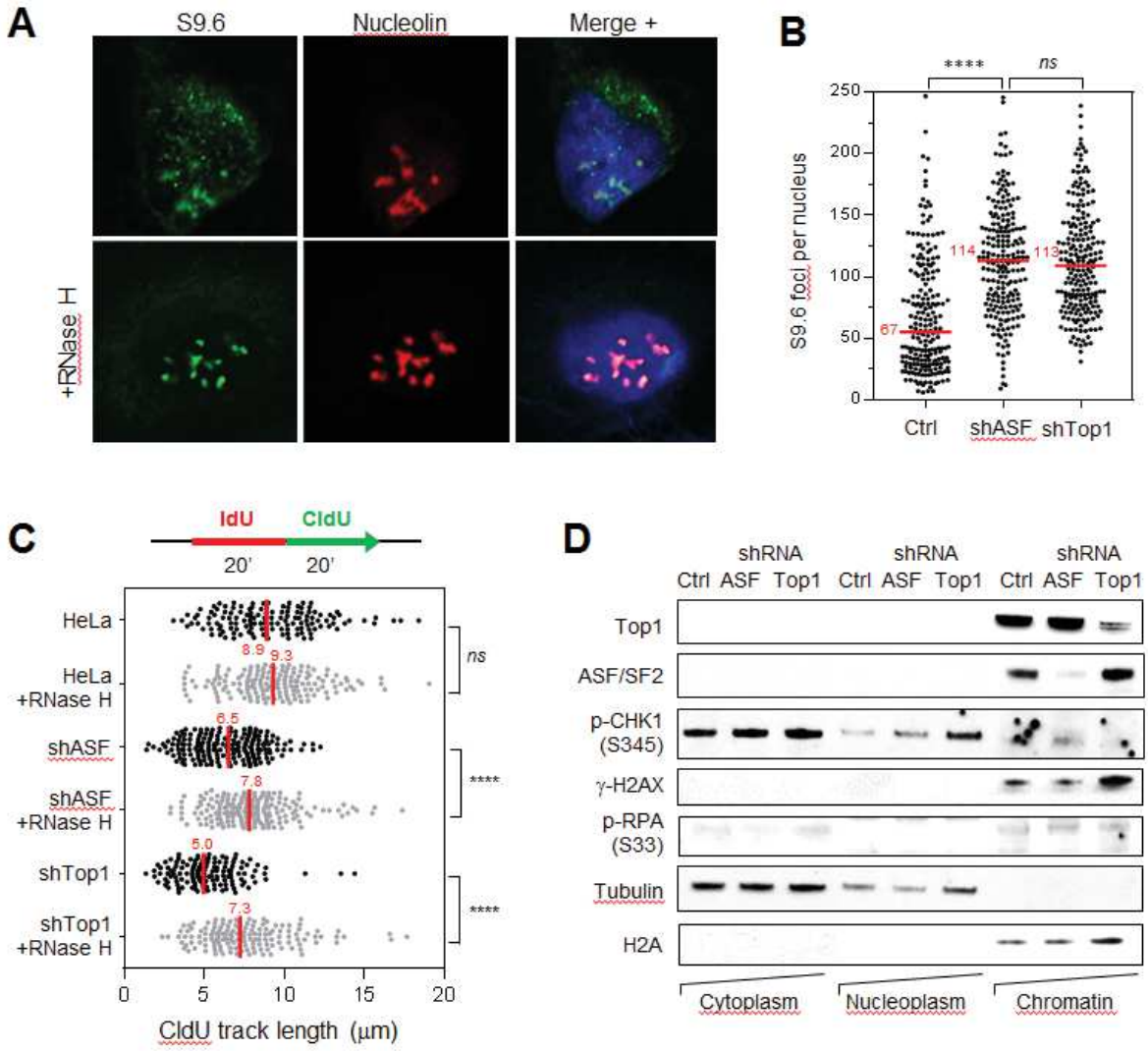

A
S9.6
Nucleolin
Merge +
+RNase H
B
****
ns
S9.6 foci per nucleus
250
200
150
100
50
0
67
114
113
Ctrl
shASF
shTop1
C
IdU
CldU
20'
20'
HeLa
HeLa +RNase H
shASF
shASF +RNase H
shTop1
shTop1 +RNase H
8.9
9.3
6.5
7.8
5.0
7.3
ns
****
****
0
5
10
15
20
CldU track length (μm)
D
shRNA
Ctrl ASF Top1 Ctrl ASF Top1 Ctrl ASF Top1
Top1
ASF/SF2
p-CHK1 (S345)
γ-H2AX
p-RPA (S33)
Tubulin
H2A
Cytoplasm
Nucleoplasm
Chromatin

FIGURE LEGENDS

Figure 1. Increased R-loop formation and RS in HeLa cells depleted for ASF/SF2 and Top1

(A) HeLa cells expressing shRNAs against ASF/SF2 (shASF) and Top1 (shTop1) for 72 hours under the control of a doxycycline-inducible promoter were seeded on coverslips and were incubated with (+RNase H) or without RNase H overnight at 37°C. RNA-DNA hybrids were detected with the S9.6 antibody (green) and nucleoli were stained with an antibody against nucleolin (red). DNA was stained with DAPI (blue). Confocal immunofluorescence microscopy was performed as described in **STAR METHODS**. Representative images are shown. Bar: 5 μm.

(B) Distribution of the number of subnuclear S9.6 foci per nucleus in HeLa, shASF and shTop1 cells. The number of foci was determined automatically for 300 cells with CellProfiler in DAPI-stained nuclei after exclusion of nucleoli. Medians are indicated in red. **** P<0.0001; *ns* not significant. Mann-Whitney rank sum test.

(C) Doxycycline-treated control, shASF and shTop1 HeLa cells were transfected for 72 hours with a mock vector (EGFP-N1) or human RNase H-EGFP (+RNase H) and were sequentially labeled with IdU and CldU for 20 minutes. Replication fork progression was measured using DNA fiber spreading as described in **STAR METHODS**. The median length of IdU tracks is indicated in red. **** P<0.0001; *ns* not significant, Mann-Whitney rank sum test.

(D) Depletion of ASF/SF2 and Top1 induces endogenous replication stress. Control, shASF and shTop1 HeLa cells were separated into chromatin and soluble fractions after doxycycline induction. The presence of replication stress markers such as phospho-CHK1 (S345), phospho-RPA32 (S33) and γ-H2AX (S139) was detected using immunoblotting.

FIGURE 2

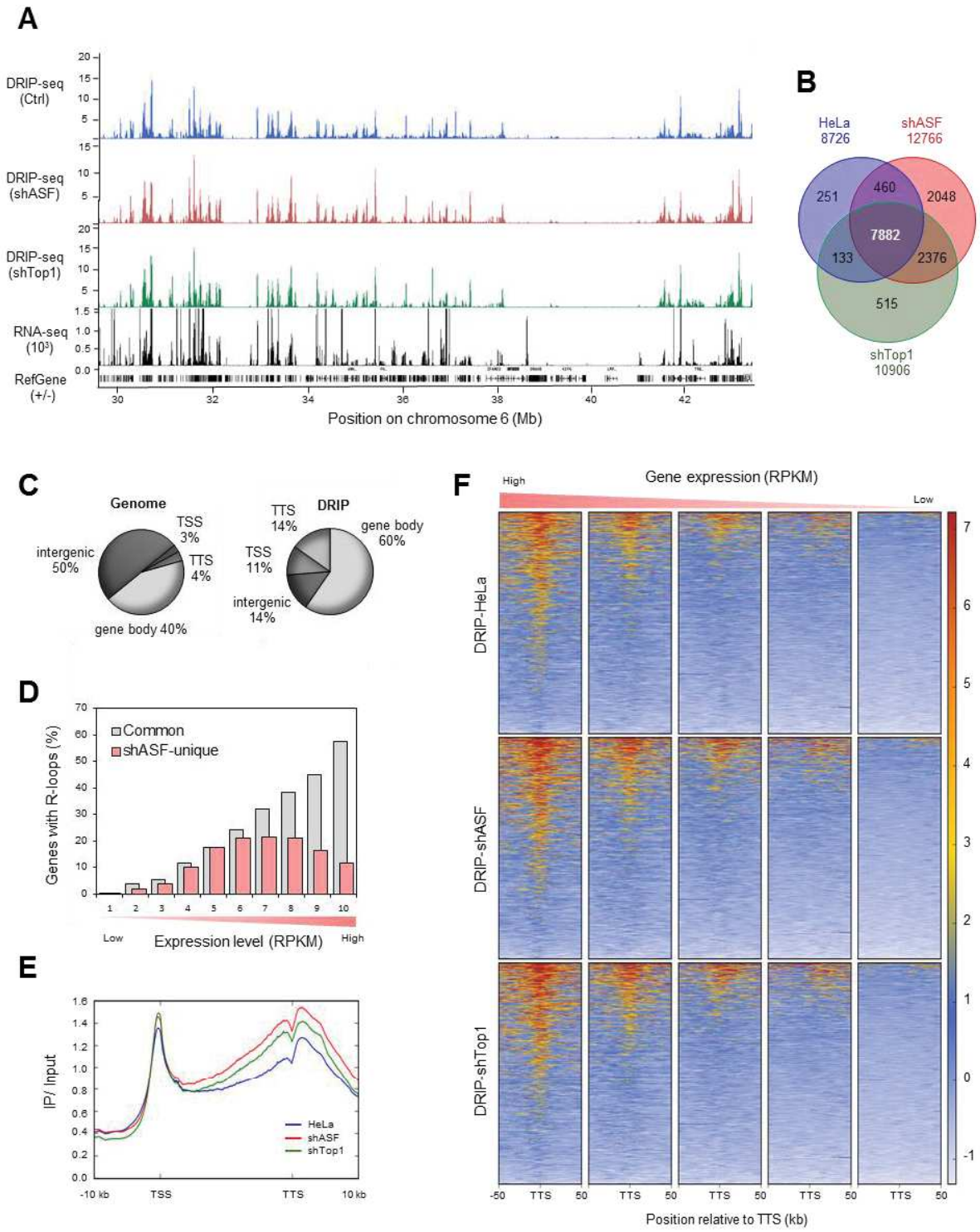

A
DRIP-seq (Ctrl)
DRIP-seq (shASF)
DRIP-seq (shTop1)
RNA-seq (10³)
RefGene (+/-)
Position on chromosome 6 (Mb)
B
HeLa 8726
shASF 12766
shTop1 10906
251
460
2048
7882
133
2376
515
C
Genome
TSS 3%
TTS 4%
gene body 40%
intergenic 50%
DRIP
TTS 14%
gene body 60%
TSS 11%
intergenic 14%
D
Common
shASF-unique
Genes with R-loops (%)
Low
Expression level (RPKM)
High
E
IP/ Input
HeLa
shASF
shTop1
-10 kb
TSS
TTS
10 kb
F
High
Gene expression (RPKM)
Low
DRIP-HeLa
DRIP-shASF
DRIP-shTop1
Position relative to TTS (kb)

Figure 2. RNA-DNA hybrids are enriched in highly expressed genes

(A) Representative region on chromosome 6 showing DRIP-seq profiles in control (blue), shASF (red) and shTop1 (green) HeLa cells. Profiles correspond to the difference between IP and input, expressed in RPKM. The RNA-seq profile (RPKM) of HeLa cells is shown (black). Gene positions of RefSeq (hg19) are also indicated.

(B) Venn diagram of the number of genes enriched in R-loops in control, shASF and shTop1 cells. R-loop positive genes correspond to genes overlapping with R-loop peaks identified with MACS2.

(C) Genomic distribution of R-loop peaks in HeLa cells. Peaks were obtained with MACS2 and were analyzed with CEAS (Cis-Regulatory Element Annotation System). The expected genome distribution (left panel) is shown for comparison. The percentage of DRIP-seq signals present in each genome region is indicated. TSS: Transcription Start Site (5'-UTR and 3 kb upstream). TTS: Transcription Termination Site (3'-UTR and 3 kb downstream).

(D) Proportion of R-loop containing genes according to gene expression levels, as determined by RNA-seq. Genes were divided into ten classes based on their level of expression (RPKM). Grey bars correspond to R-loop positive genes present in both control and shASF cells. Red bars represent genes showing R-loop peaks specifically in shASF cells.

(E) Metaplot of the distribution of RNA-DNA hybrid signals (IP/input) along human genes in control (blue), shASF (red) and shTop1 (green) HeLa cells. TSS: Transcription Start Site. TTS: Transcription Termination Site.

(F) Heatmap of the intensity of RNA-DNA hybrid signals at TTS of genes organized in five classes with increasing levels of expression (RNA-seq) of shTop1 cells.

FIGURE 3

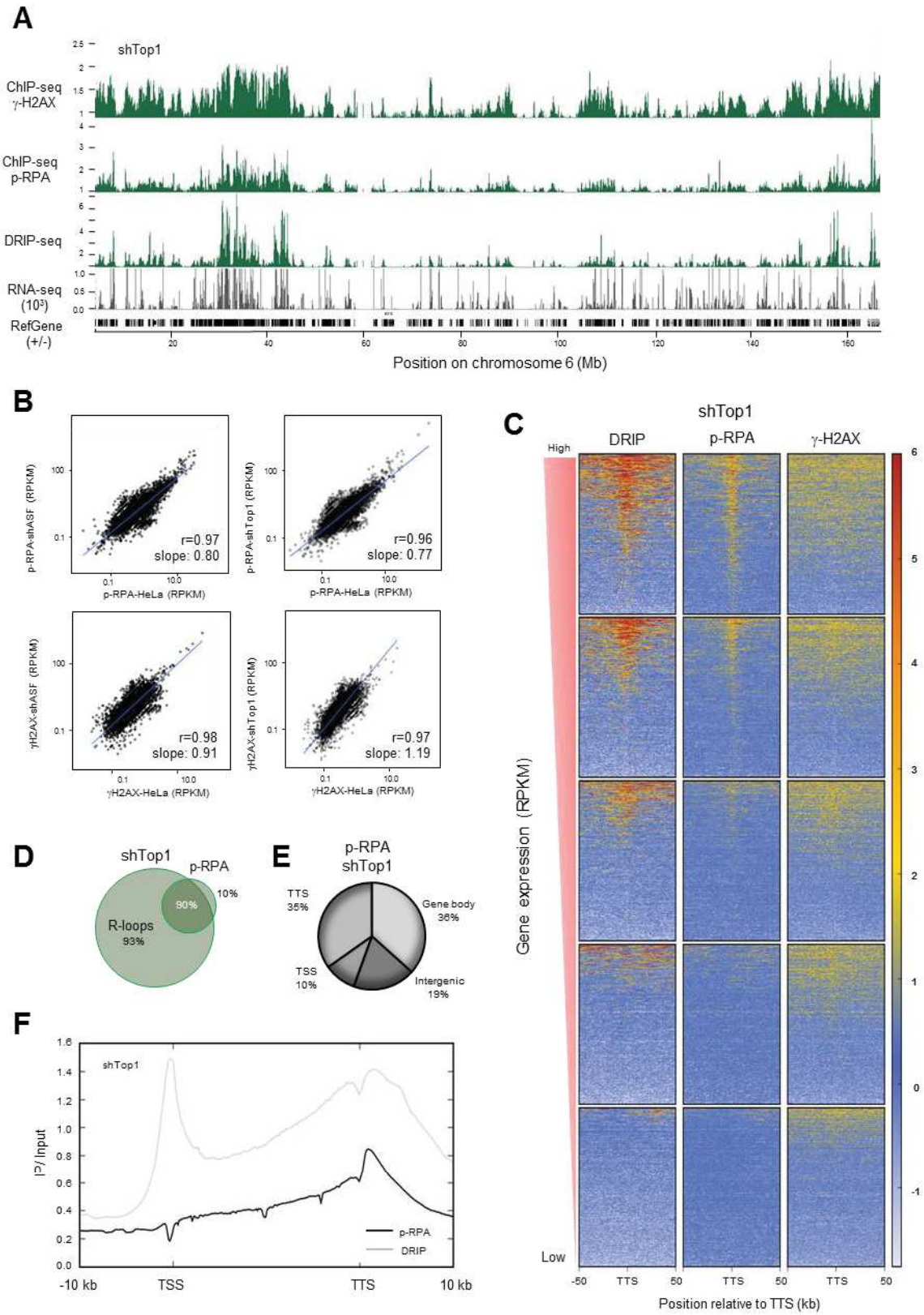

A
shTop1
ChIP-seq γ-H2AX
ChIP-seq p-RPA
DRIP-seq
RNA-seq (10³)
RefGene (+/-)
Position on chromosome 6 (Mb)
B
p-RPA-shASF (RPKM)
p-RPA-HeLa (RPKM)
r=0.97
slope: 0.80
p-RPA-shTop1 (RPKM)
r=0.96
slope: 0.77
γH2AX-shASF (RPKM)
γH2AX-HeLa (RPKM)
r=0.98
slope: 0.91
γH2AX-shTop1 (RPKM)
r=0.97
slope: 1.19
C
shTop1
DRIP
p-RPA
γ-H2AX
High
Low
Gene expression (RPKM)
Position relative to TTS (kb)
D
shTop1
p-RPA
90%
10%
R-loops
93%
E
p-RPA
shTop1
TTS
35%
Gene body
36%
TSS
10%
Intergenic
19%
F
shTop1
p-RPA
DRIP
-10 kb
TSS
TTS
10 kb

Figure 3. Differential enrichment of replication stress marks at R-loop enriched regions

(A) Distribution of γ-H2AX, p-RPA32 (S33) and RNA-DNA hybrids signals along a representative region on chromosome 6 in shTop1 cells. Enrichment is determined as the difference between IP and input and is expressed in RPKM. The RNA-seq profile of shTop1 cells is shown in black. Gene positions of RefSeq (hg19) are also indicated.

(B) Scatter plots of the intensity of p-RPA and γ-H2AX signals in human genes in shASF or shTop1 cells relative to control HeLa cells. Correlation coefficients and slopes of linear regressions are indicated.

(C) Heatmap of the intensity of RNA-DNA hybrids (DRIP), p-RPA and γ-H2AX signals at TTS in five groups of genes with increased expression levels (RNA-seq). Genes were sorted relative to the intensity of DRIP signal.

(D) Venn diagram of the number of genes overlapping with R-loop and p-RPA peaks (MACS2) in shTop1 cells. The percentages of overlapping and non-overlapping genes are indicated.

(E) Genomic distribution of p-RPA peaks in shTop1 cells. ChIP-seq peaks were obtained with MACS2 and analyzed with CEAS.

(F) Metaplot of the intensity of RNA-DNA hybrids and p-RPA along human genes in shTop1 cells.

FIGURE 4

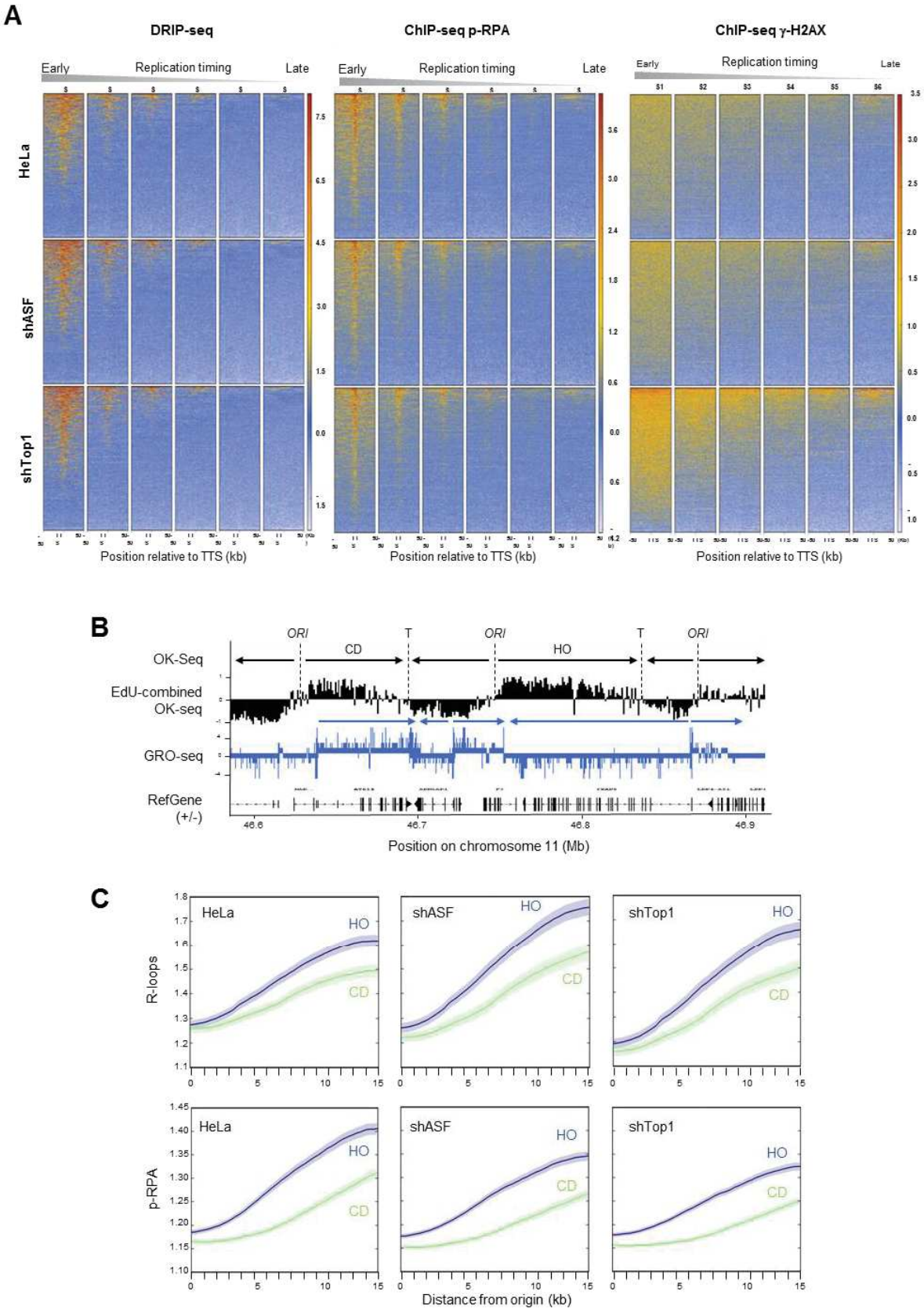

A
DRIP-seq
ChIP-seq p-RPA
ChIP-seq γ-H2AX
Early
Replication timing
Late
HeLa
shASF
shTop1
Position relative to TTS (kb)
B
ORI
CD
T
HO
OK-Seq
EdU-combined OK-seq
GRO-seq
RefGene (+/-)
46.6
46.7
46.8
46.9
Position on chromosome 11 (Mb)
C
HeLa
shASF
shTop1
HO
CD
R-loops
p-RPA
Distance from origin (kb)

Figure 4. R-loops are preferentially formed in early replicating genes with head-on collisions

(A) Heatmap of the intensity of DNA-RNA hybrids, p-RPA and γ-H2AX signals on the TTS of six classes of genes sorted according to their time of replication. Replication timing from S1 (early S) to S6 (late S) is derived from published Repli-seq data (Hansen et al., 2010).

(B) Representative region on human chromosome 11 showing the position of replication initiation (ORI) and termination (T) sites and the direction of fork progression (black), as determined by OK-seq (Petryk et al., 2016). The direction of transcription (blue) is determined from published GRO-seq data.

(C) Mean profiles of R-loop and p-RPA enrichment in genomic regions showing a preferential head-on (HO) and codirectional (CD) orientation of genes relative to the direction of fork progression, as determined from OK-seq and GRO-seq datasets.

FIGURE S1

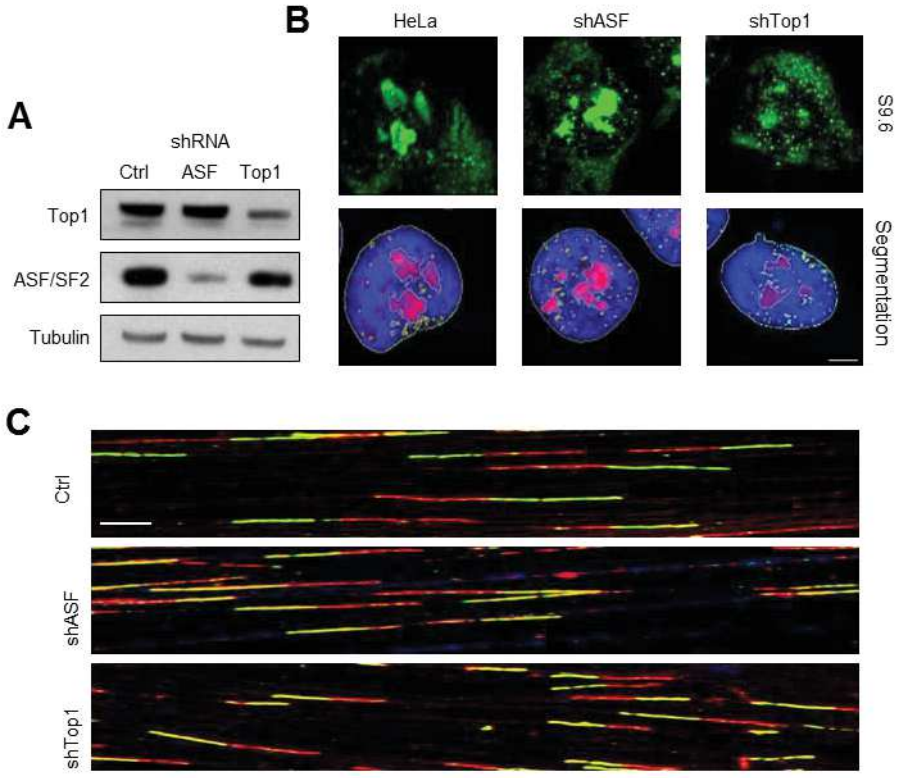

A
shRNA
Ctrl
ASF
Top1
Top1
ASF/SF2
Tubulin
B
HeLa
shASF
shTop1
S9.6
Segmentation
C
Ctrl
shASF
shTop1

Figure S1. Depletion of ASF/SF2 and Top1 in HeLa cells increases the accumulation of RNA-DNA hybrids

(A) Expression levels of ASF/SF2 and Top1 in HeLa cells. HeLa cells transduced by inducible lentiviral virions expressing shRNAs against ASF/SF2 or Top1 were treated with doxycycline for 72 hours. The expression of ASF/SF2 and Top1 was detected by Western blotting using specific antibodies.

(B) Control, shASF and shTop1 cells were treated with doxycycline for 72 hours and seeded on the coverslips. The presence of RNA-DNA hybrids was detected by confocal immunofluorescence microscopy as described in Figure 1. Representative images of RNA-DNA hybrids (green) and examples of segmentation obtained with CellProfiler are shown. DAPI: blue, nucleolin: red, RNA-DNA hybrids: green. Bar: 3 μm.

(C) Representative images of individual DNA fibers showing IdU (red) and CldU (green) tracks in control, shASF and shTop1 cells are shown. Bar: 5 μm.

FIGURE S2

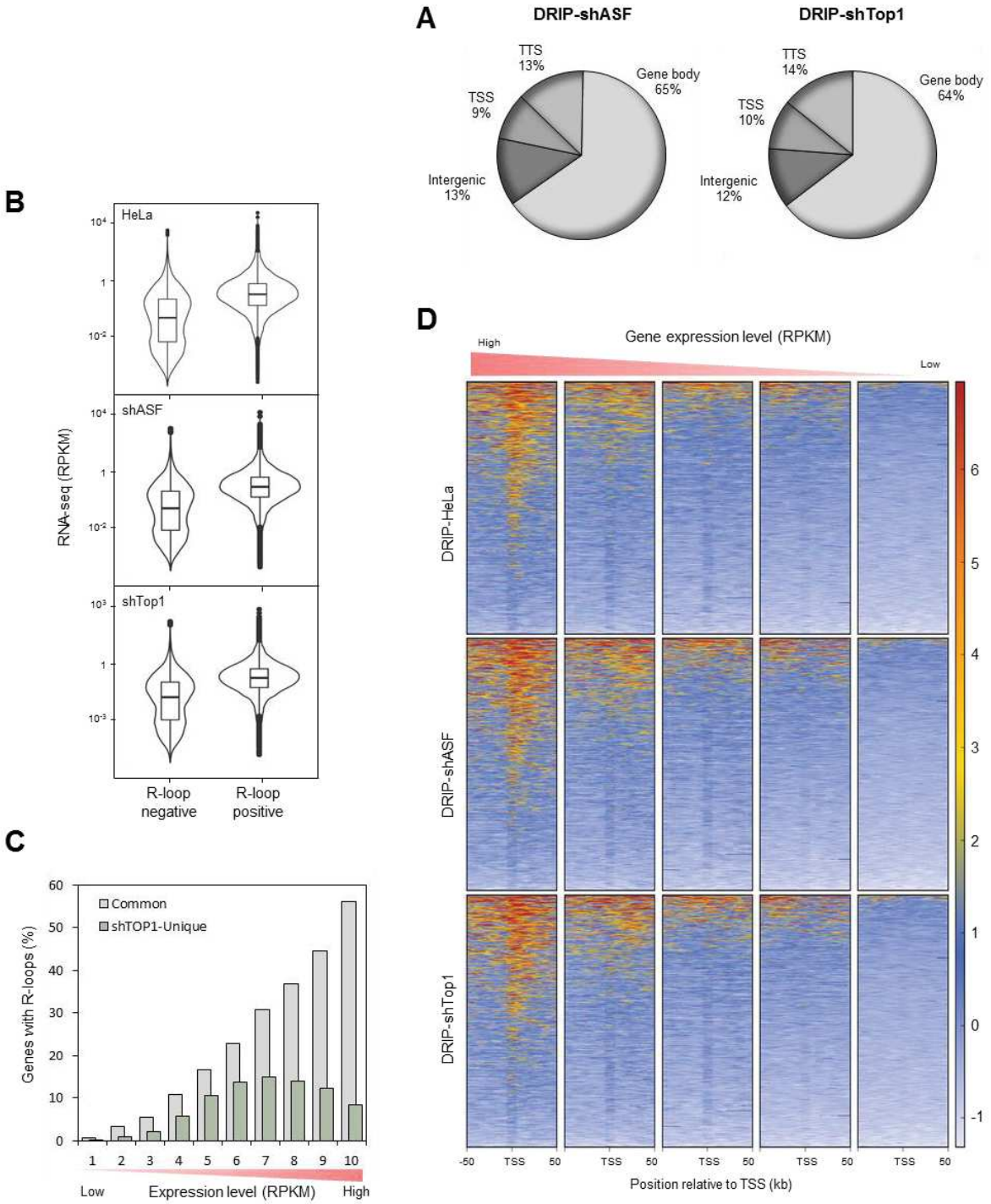

A
DRIP-shASF
TTS 13%
Gene body 65%
TSS 9%
Intergenic 13%
DRIP-shTop1
TTS 14%
Gene body 64%
TSS 10%
Intergenic 12%
B
HeLa
shASF
shTop1
RNA-seq (RPKM)
R-loop negative
R-loop positive
C
Common
shTOP1-Unique
Genes with R-loops (%)
Low
Expression level (RPKM)
High
D
Gene expression level (RPKM)
High
Low
DRIP-HeLa
DRIP-shASF
DRIP-shTop1
TSS
Position relative to TSS (kb)

Figure S2. R-loops are enriched in highly expressed genes

(A) Genomic distribution of R-loop peaks (MACS2) in shASF and shTop1 cells.

(B) Violin-plots of RNA-seq (RPKM) in genes containing or not R-loop signals.

(C) Proportion of R-loop containing genes in classes of genes with increased transcription, as determined by RNA-seq. Genes were divided into ten classes based on their expression levels (RPKM). Light grey bars correspond to R-loop positive genes present in both control and shTop1 cells. Green bars represent genes showing R-loop peaks specifically in shTop1 cells.

(D) Heatmap of the intensity of RNA-DNA hybrid signals at TSS in genes organized in five classes with increasing levels of expression (RNA-seq) in control, shASF and shTop1 cells.

FIGURE S3

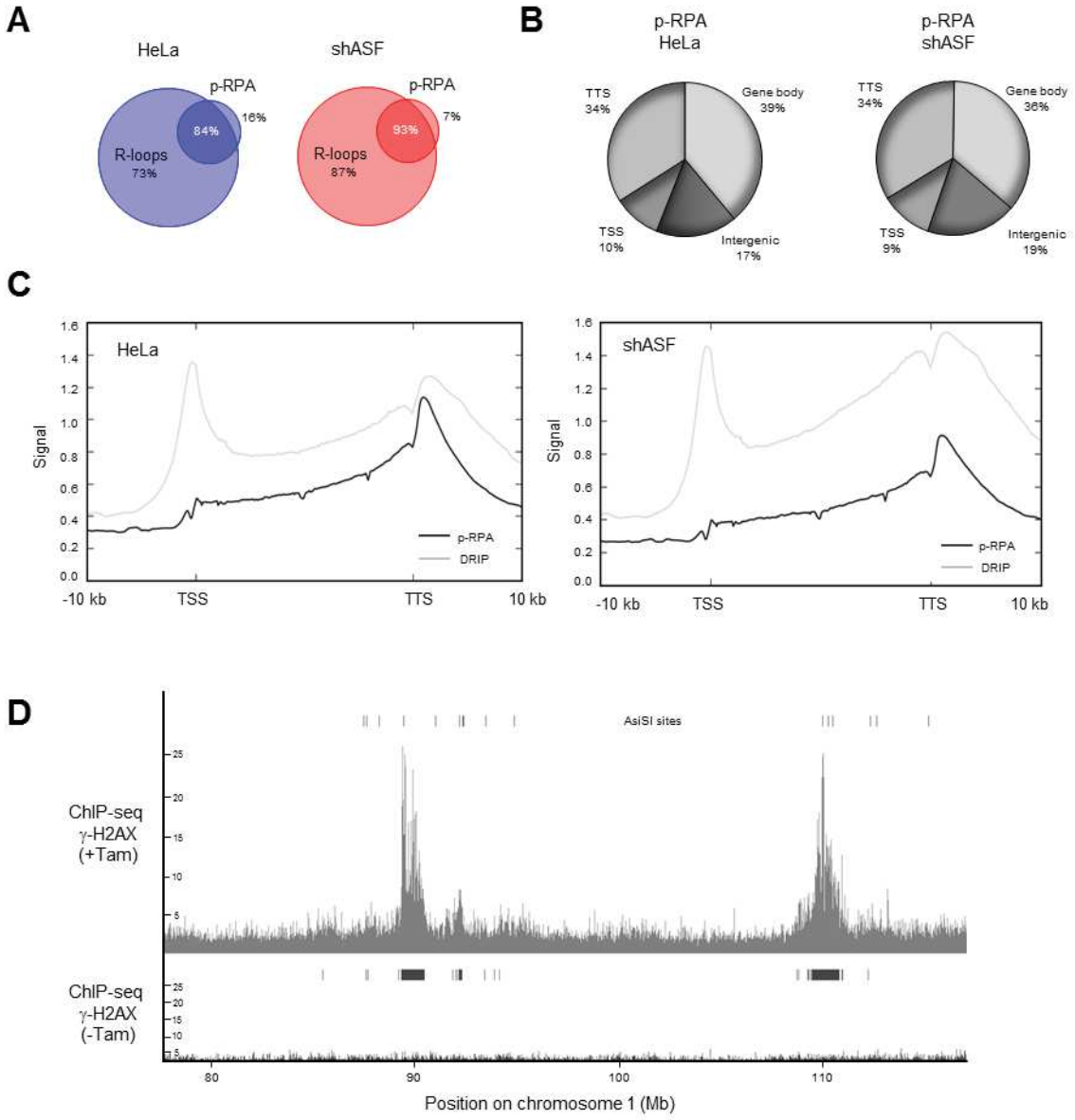

A
HeLa
p-RPA
16%
84%
R-loops
73%
shASF
p-RPA
7%
93%
R-loops
87%
B
p-RPA
HeLa
TTS
34%
Gene body
39%
TSS
10%
Intergenic
17%
p-RPA
shASF
TTS
34%
Gene body
36%
TSS
9%
Intergenic
19%
C
HeLa
Signal
p-RPA
DRIP
-10 kb
TSS
TTS
10 kb
shASF
Signal
p-RPA
DRIP
-10 kb
TSS
TTS
10 kb
D
AsiSI sites
ChIP-seq
γ-H2AX
(+Tam)
ChIP-seq
γ-H2AX
(-Tam)
Position on chromosome 1 (Mb)

Figure S3. Replication stress correlates with R-loop formation

(A) Venn diagram of the number of genes containing R-loop and p-RPA signals in control and shASF cells.

(B) Genomic distribution of p-RPA in control and shASF cells.

(C) Metaplot of the intensity of RNA-DNA hybrids and p-RPA along human genes in control and shASF cells.

(D) Representative region on chromosome 1 showing the accumulation of γ-H2AX signal around AsiS1 cut sites.

(E) Heatmap of the intensity of RNA-DNA hybrid signals at TTS of genes organized in five classes with increasing levels of expression (RNA-seq) of control and shASF cells.

(F) Heatmap of the intensity of RNA-DNA hybrid signals at TSS of genes organized in five classes with increasing levels of expression (RNA-seq) of control, shASF and shTop1 cells.

FIGURE S3

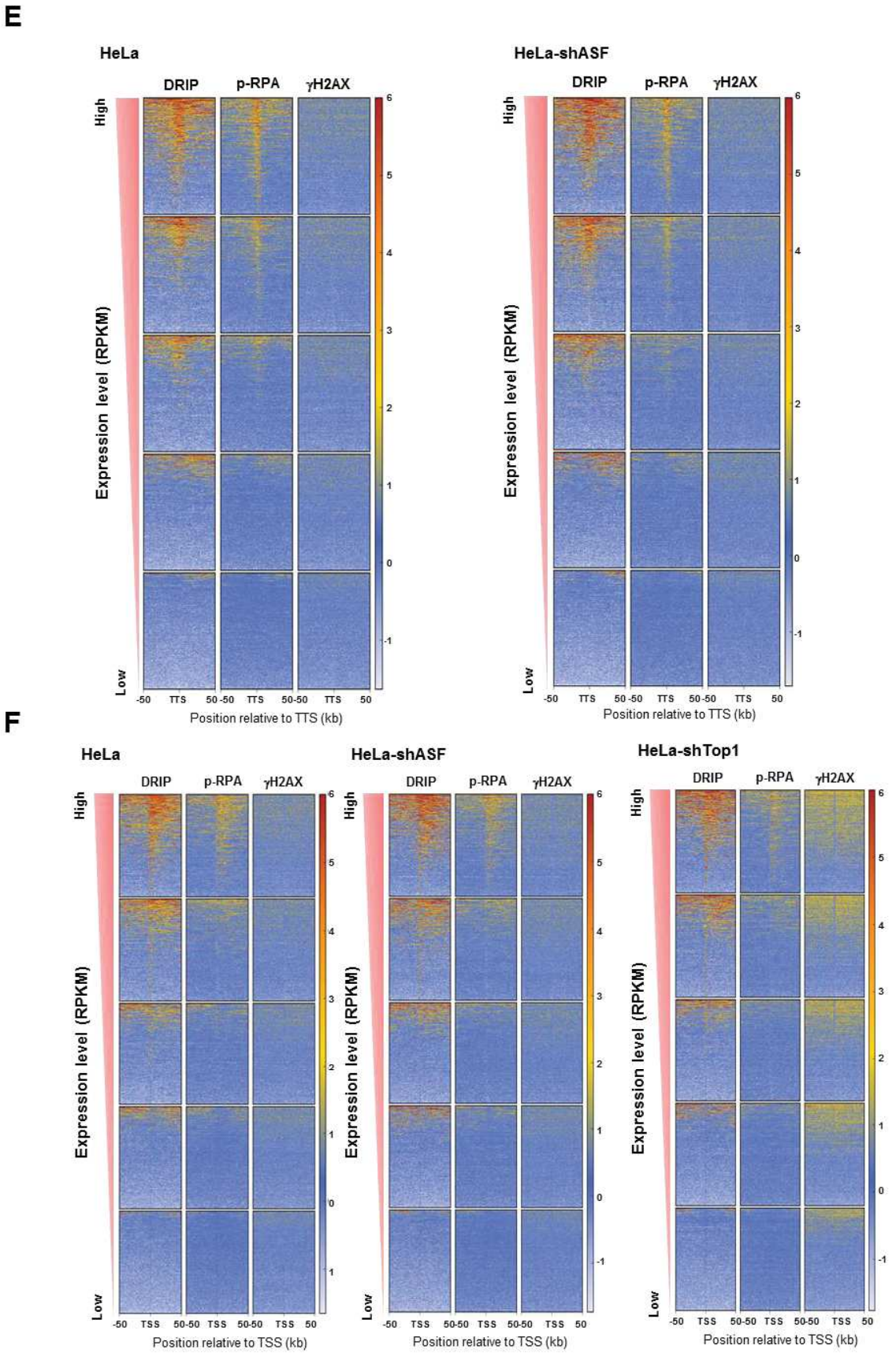
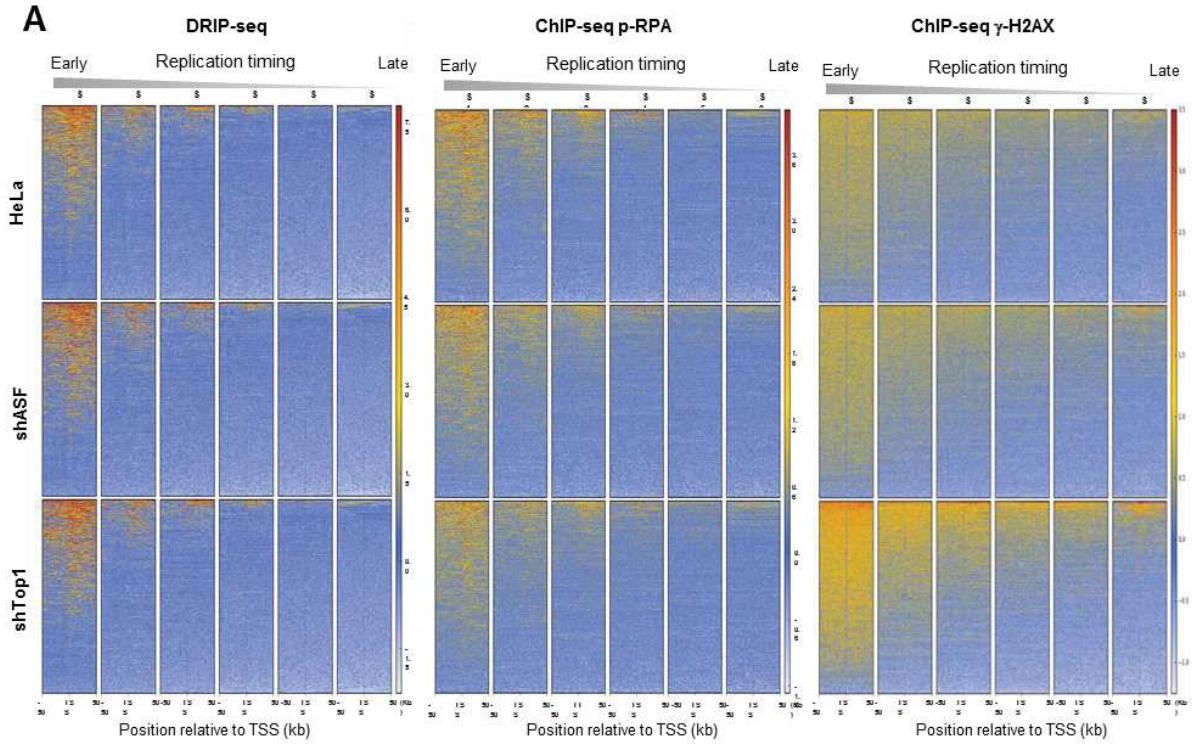

E
HeLa
DRIP
p-RPA
γH2AX
High
Low
Expression level (RPKM)
-50 TTS 50 -50 TTS 50 -50 TTS 50
Position relative to TTS (kb)
HeLa-shASF
DRIP
p-RPA
γH2AX
High
Low
Expression level (RPKM)
-50 TTS 50 -50 TTS 50 -50 TTS 50
Position relative to TTS (kb)
F
HeLa
DRIP
p-RPA
γH2AX
High
Low
Expression level (RPKM)
-50 TSS 50-50 TSS 50-50 TSS 50
Position relative to TSS (kb)
HeLa-shASF
DRIP
p-RPA
γH2AX
High
Low
Expression level (RPKM)
-50 TSS 50-50 TSS 50-50 TSS 50
Position relative to TSS (kb)
HeLa-shTop1
DRIP
p-RPA
γH2AX
High
Low
Expression level (RPKM)
-50 TSS 50-50 TSS 50-50 TSS 50
Position relative to TSS (kb)

Figure S4. Replication stress is correlated with R-loop formation during replication.

Heatmap presentation of DRIP-seq signals on Transcription Termination Sites (TTS) plus and minus 50 kb of genes classified according to the replication timing. Classes (early to late, S1; S2; S3; S4; S5; S6) were built using RepliSeq data (ENCODE GSM923449).



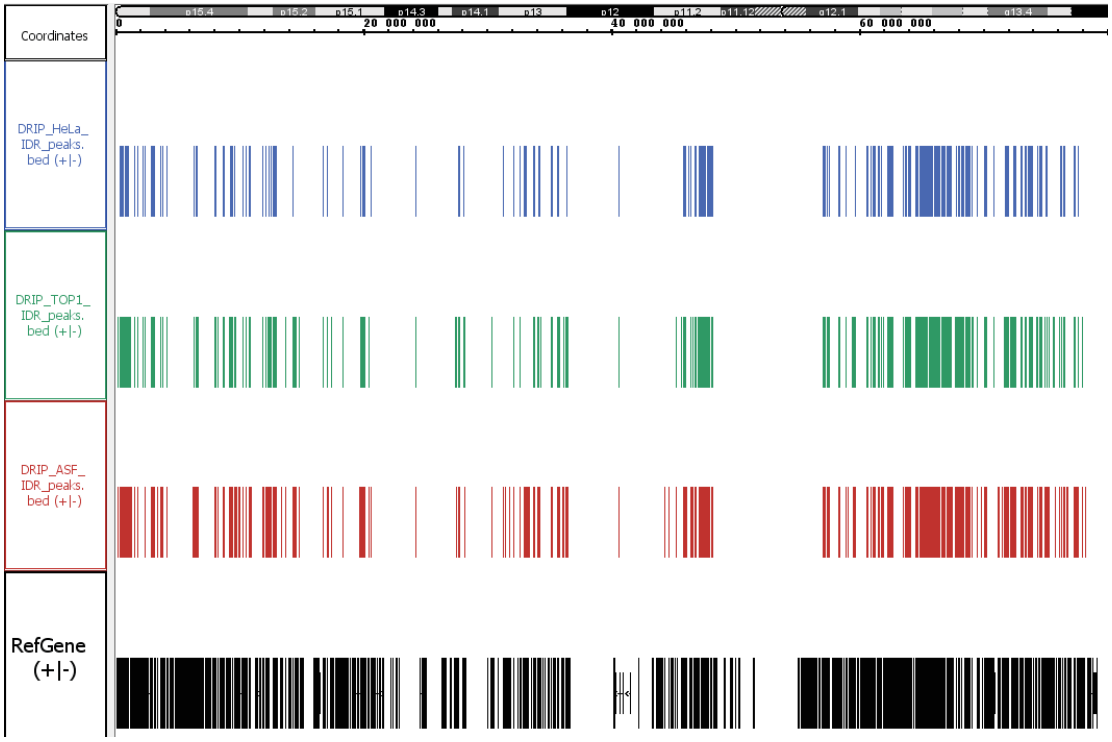


Figure 26 : Résultat du peak calling du DRIP-Seq : Vue large sur le début du chromosome 11 (0-80 Mb). Les résultats sont présentés pour nos trois lignées cellulaires HeLa (bleu), shTOP1 (vert) et shASF (rouge). La distribution des gènes est également montrée (noir). Les pics ont été obtenus avec le logiciel MACS2, puis sélectionnés avec la méthode IDR.

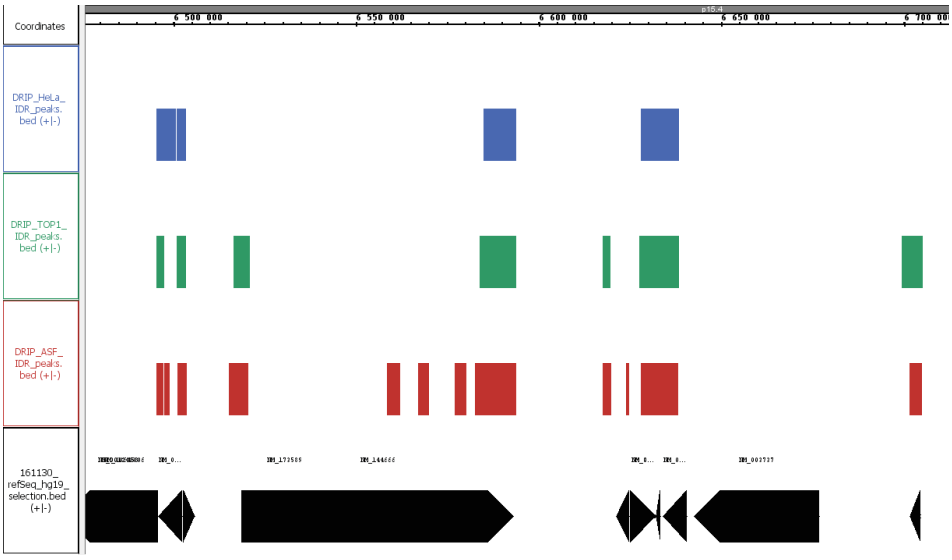


Figure 27 : Résultat du peak calling du DRIP-Seq : Zoom sur une portion du chromosome 11. Les résultats sont présentés pour nos trois lignées cellulaires HeLa (bleu), shTOP1 (vert) et shASF (rouge). La distribution des gènes est également montrée (noir). Les pics ont été obtenus avec le logiciel MACS2, puis sélectionnés avec la méthode IDR.

2. Analyse des données de DRIP-Seq

La première étape de mon projet a été d'établir une carte de la distribution génomique des R-loops dans nos trois lignées cellulaires. Ces informations ont servi à comparer la distribution des hybrides dans nos cellules aux signaux de stress obtenus par ChIP-Seq (pRPA, γ-H2AX), mais également avec de nombreuses annotations publiées pour les cellules HeLa (Gènes, marque d'histone, Repli-Seq, OK-Seq…). J'ai décidé d'utiliser la version hg19 du génome humain, disponible sur le site de UCSC, car la majorité des résultats disponibles sont alignés sur cette version (http://hgdownload.soe.ucsc.edu/downloads.html). Pour comparer mes données aux annotations de gènes, j'ai utilisé la référence refSeq, disponible également sur le site de UCSC (https://genome.ucsc.edu/cgi-bin/hgTables).

2.1. La recherche de pics

La recherche de pics de DRIP-Seq peut se faire avec un logiciel cherchant des pics étroits (Ginno et al. 2012) tel que MACS2 (Zhang et al. 2008). Comme nos données de DRIP-Seq ont été répliquées, j'ai utilisé une p-value souple (0.05) pour obtenir un grand nombre de pics me permettant d'appliquer l'algorithme IDR (Voir intro ChIP-Seq) pour sélectionner les pics reproductibles. J'ai obtenu 8 483 positions enrichies pour le contrôle HeLa, 12883 pour shTOP1 et 19 598 pour shASF/SF2.

2.2. Analyse et annotation des R-loops sur le génome

La distribution des pics est similaire pour les trois échantillons (Fig. 26). Une partie des nouvelles positions enrichies pour les échantillons shASF/SF2 et shTOP1 se forme autour de positions déjà marquées dans les cellules HeLa (Fig. 27), indiquant un étalement du signal à ces positions.

On constate également que le signal est corrélé avec la distribution des gènes (Fig. 26). Pour confirmer cette impression, j'ai utilisé le logiciel bedTools (Quinlan 2002) pour croiser mes pics avec l'annotation du génome (RefSeq). Environ 85% du signal se trouve sur un gène dans les trois lignées cellulaires. Les annotations formant des R-loops dans les cellules HeLa ont aussi un signal dans les cellules shASF, cela représente 65% des gènes marqués dans cette lignée.

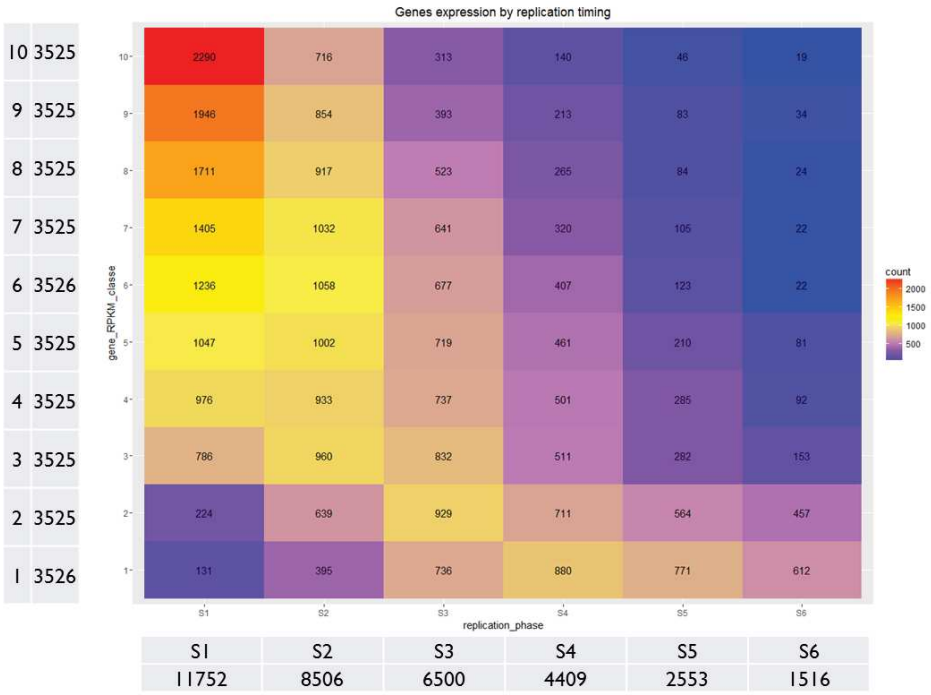


Figure 28 : Distribution des gènes en fonction de leur niveau d'expression et de leur timing de réplication. Le timing de réplication est divisé en 6 phases de la plus précoce à la plus tardive (S1-S6). Les gènes ont été séparés en 10 en fonction leur de niveau d'expression, du plus faible au plus fort (1-10). Les couleurs sont indicatives du nombre de gènes par catégorie.

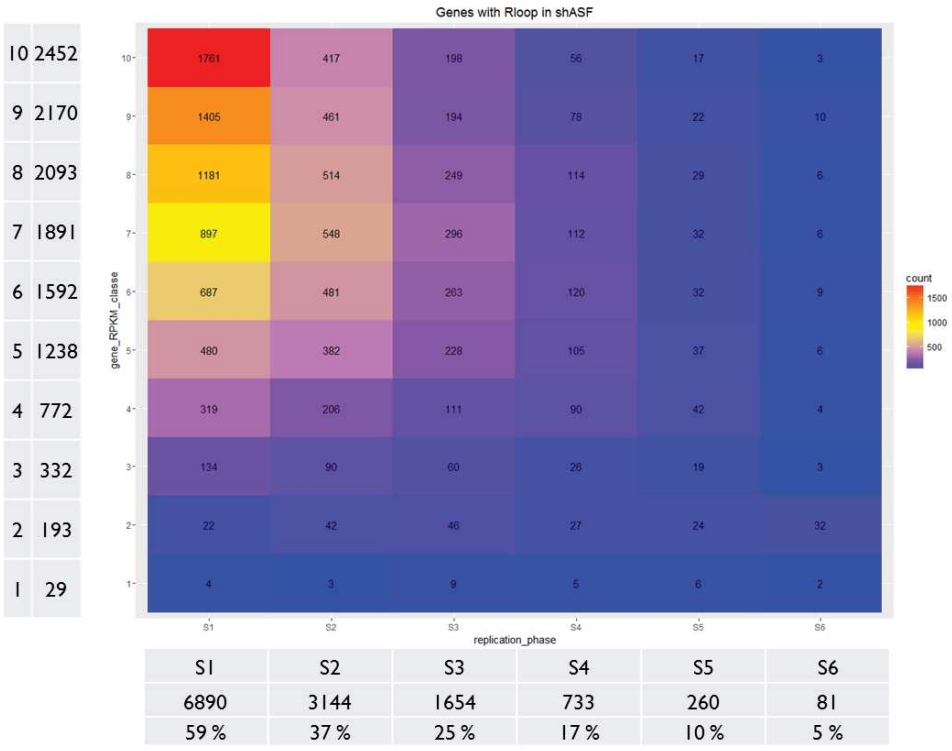


Figure 29: Distribution des gènes formant des R-loops en fonction de leur niveau d'expression et de leur timing de réplication. Dans le tableau (en bas), le pourcentage indique la proportion de gènes, par timing de réplication, formant des R-loops.

Le logiciel CEAS (Cis-regulatory Element Annotation System, Shin et al. 2009) analyse la distribution d'un signal par rapport l'annotation des gènes. Ces résultats confirment que les R-loops sont enrichies sur les gènes et plus particulièrement aux extrémités (Fig. 2 Publication). La distribution est similaire dans le contrôle et dans nos lignées shASF et shTOP1. Cela veut dire que favoriser la formation des R-loops n'entraine pas leur apparition spontanée sur n'importe quelle position du génome, mais sur le corps des gènes, principalement en 3' ou 5', à des positions sous contrôle dans une cellule saine.

Nos données de RNA-Seq et de timing de réplication ont révélé que les R-loops se formaient majoritairement dans les gènes avec une forte expression et répliqués précocement. La déplétion d'ASF/SF2 favorise la formation de R-loops sur des gènes dont le niveau d'expression est moins élevé (Fig. 2 Publication). Cependant, comme la majorité des gènes fortement ou moyennement transcrits sont répliqués précocement (Fig. 28), le timing de réplication des gènes formant des R-loops reste précoce (Fig. 29).

3. Analyse des données de ChIP-Seq pour p-RPA S33

Pour visualiser si le ralentissement des fourches observées dans nos lignées cellulaires est dû à une interaction directe entre les R-loops et la fourche de réplication, nous avons réalisé un ChIP-Seq de pRPA-S33 qui marque le ralentissement et l'arrêt des fourches.

3.1. La recherche de pics

L'analyse de notre séquençage avec le logiciel MACS2 nous a permis d'obtenir 11 137 pics pour la lignée contrôle, 3 897 pour les cellules shASF/SF2 et 2 875 pour l'échantillon shTOP1. Comme pour le DRIP-Seq, le croisement avec l'annotation des gènes montre un fort enrichissement du pRPA sur les gènes, environ 60% du signal. Et si l'on augmente la recherche de 10 kb autour des gènes, on trouve que 90% du signal est proche des gènes. La majorité des gènes touchés (80%) sont ceux formant des R-loops, comme indiqué dans notre article.

Le fait de trouver moins de pics dans nos lignées modifiées pourrait s'expliquer par le fait que l'initiation de nouvelles fourches est inhibée en cas de stress réplicatif. En effet, les 11 137 pics trouvés dans les cellules HeLa ne couvrent que 3 710 gènes, beaucoup d'entre eux ayant de nombreux pics sur leur corps (Fig 30). 2 435 gènes ont un signal dans la lignée shASF/SF2 et 90% de ces gènes ont également un signal dans la lignée contrôle. On constate

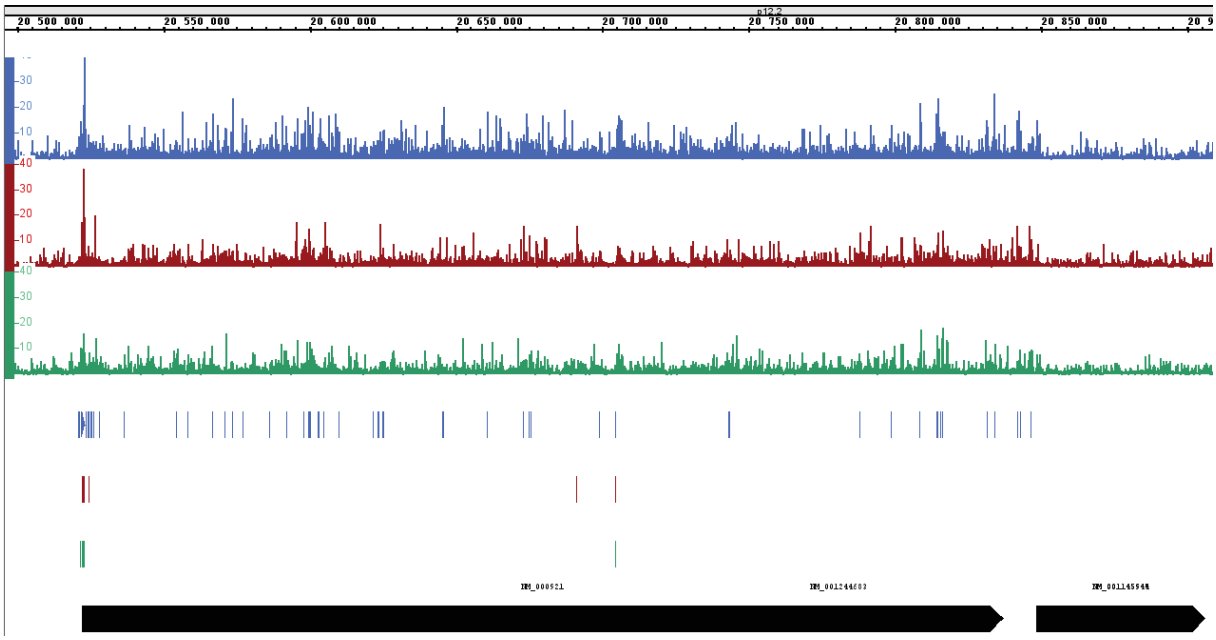


Figure 30: Résultat du peak calling du ChIP-Seq de pRPA : Zoom sur le gène PDE3A. Résultats présentés pour nos trois lignées cellulaires HeLa (bleu), shTOP1 (vert) et shASF (rouge). Le profil du signal est présenté en haut, puis les résultats du peak-calling en bas.

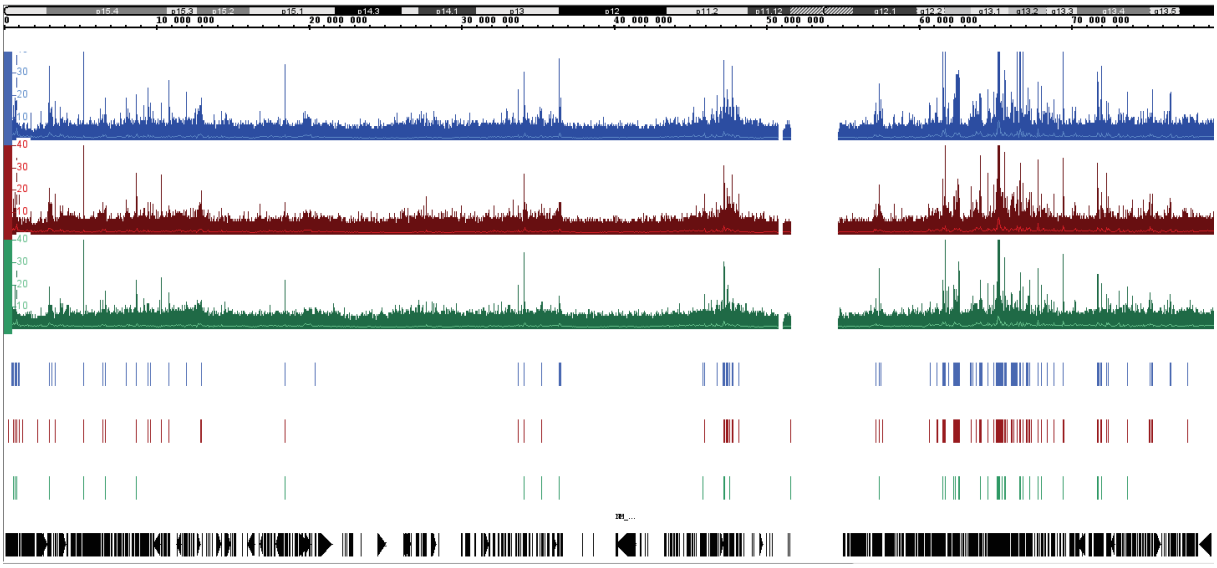


Figure 31: Résultat du peak calling du ChIP-Seq de pRPA : Vue large d'une portion du chromosome 11. Résultats présentés pour nos trois lignées cellulaires HeLa (bleu), shTOP1 (vert) et shASF (rouge). Le profil du signal est présenté en haut, puis les résultats du peak-calling en bas.

aussi que bien que le résultat du peak calling diffère entre les lignées, les profils obtenus sont très similaires (Fig. 31).

L'absence de réplique des données ne permet pas de confirmer les informations trouvées lors de l'analyse et d'affiner le peak calling. Malheureusement, l'anticorps utilisé pour la capture du pRPA n'a pas fonctionné lorsque nous avons répété le ChIP-Seq. Mais la vérification de l'intensité du signal sur une sélection de gènes, avec la technique de qPCR (quantitative PCR), a confirmé le fait que le signal de pRPA était plus intense dans la lignée contrôle et plus faible dans la lignée shTOP1.

4. Traitement des données de γ-H2AX

Comme indiqué précédemment, le signal γ-H2AX peut s'étendre sur plusieurs mégabases suite à une cassure chromosomique. Il diffère en cela de la plupart des autres marques d'histones ou des facteurs de transcription habituellement étudiés avec le ChIP-Seq. Ce signal s'étend de façon asymétrique des deux côtés de la cassure, les gènes activement transcrits pouvant modifier la distribution du signal (Iacovoni et al. 2010). Je vais décrire dans ce chapitre les différentes étapes de l'analyse de ce signal et les résultats obtenus en relation avec les R-loops

4.1. Le modèle cellulaire AsiSI-U2OS-ER

Pour vérifier l'efficacité de la technique de ChIP-Seq de γ-H2AX pour la détection des cassures double brin de l'ADN, nous avons utilisé une lignée cellulaire dans laquelle des cassures double brin peuvent être induites au niveau de sites spécifiques.

En 2010 le laboratoire du docteur Legube a publié une étude de la répartition du signal de de γ-H2AX autour des DSB dans des cellules de mammifères (Iacovoni et al. 2010). Pour cela, ils ont créé une lignée cellulaire (AsiSI–ER-U2OS) dans laquelle l'enzyme de restriction AsiSI est utilisée pour induire des cassures double brin. L'enzyme est fusionnée avec un récepteur à œstrogène modifié qui entraine la translocation de l'enzyme dans le noyau en présence de 4-hydroxy tamoxifene (4OHT). De cette façon, l'induction des cassures est contrôlée par le traitement des cellules au 4OHT. Les sites de coupures étant connus sur le génome humain, il est facile de vérifier que celles-ci induisent un marquage par γ-H2AX et la répartition de ce signal.

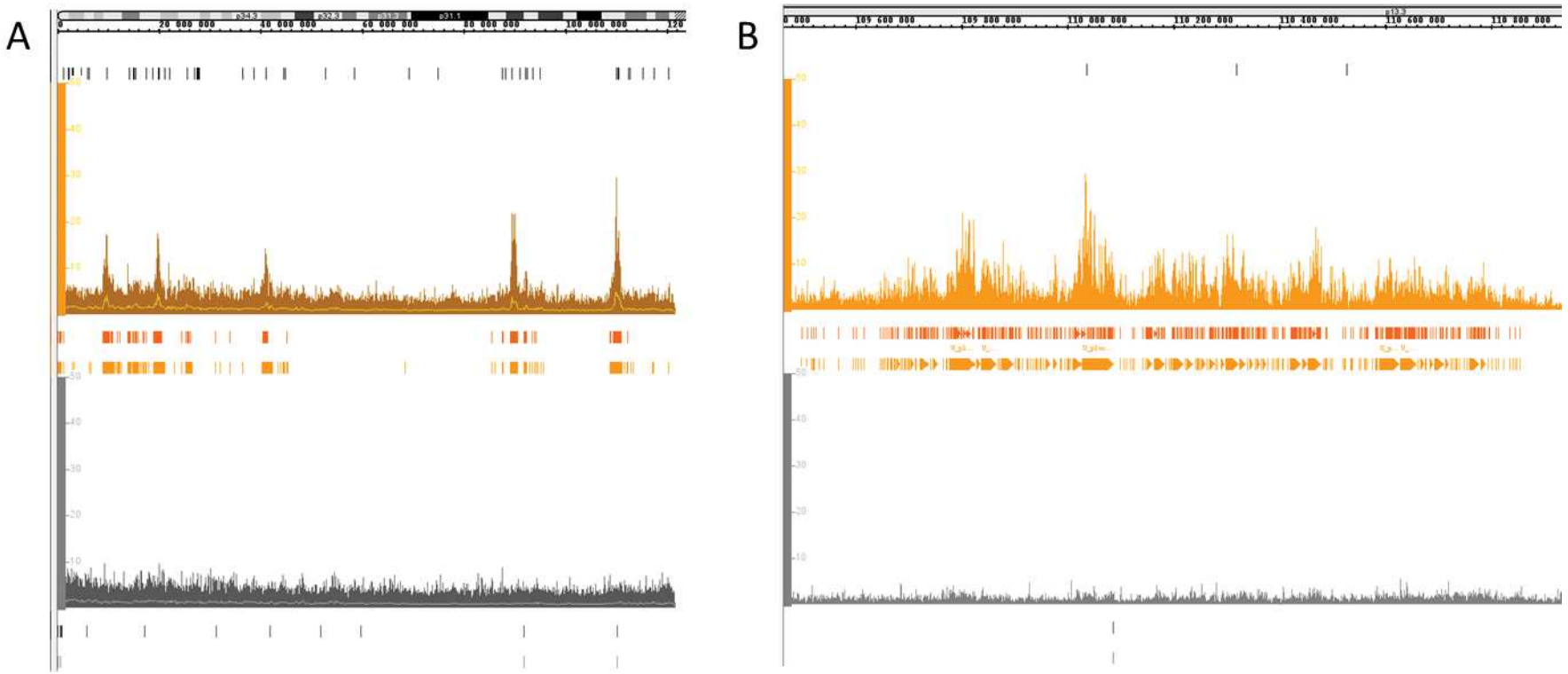


Figure 32 : Profil ChIP-Seq γ-H2AX sur les cellules U2OS. (A) Profil sur le chromosome 1. (B) Zoom sur un groupe de coupure AsiSI. De haut en bas. En noir les sites de coupure de l'enzyme AsiSI. En orange, le profil des cellules avec cassures induites. En orange clair et foncé sont représentés les pics appelés par MACS2 narrow et broad. En gris, les mêmes résultats sont présentés sur l'échantillon contrôle.

L'analyse visuelle d'une région représentative du génome montre la présence d'un fort signal couvrant plusieurs dizaines de kilobases autour des sites de coupures (Fig. 32). Lorsque l'on regarde de plus près la répartition du γ-H2AX, on s'aperçoit que le signal n'est pas parfaitement centré sur la coupure et qu'il est impossible, si deux coupures sont proches, de déterminer si une seule est active ou bien les deux (Fig 32). On détecte également un niveau de bruit de fond assez élevé, lié à la présence de stress génotoxique spontané dans les cellules tumorales. Néanmoins, ces résultats montrent que la technique permet de détecter des cassures double brin.

4.1.1. Analyse des données γ-H2AX dans les cellules U2OS

La méthode recommandée pour déterminer la position de régions enrichies en un facteur donné par ChIP-Seq est le « peak calling ». Comme j'utilise MACS2 pour les données de DRIP-Seq et que je souhaitais conserver une cohérence dans les méthodes d'analyses, j'ai analysé les profils ChIP-Seq de γ-H2AX avec ce logiciel. Le résultat du peak calling n'est pas concluant. MACS2 est recommandé pour l'analyse de facteurs de transcription qui donnent un signal peu étalé sur le génome, mais il existe une option (broad) pour appeler des pics plus larges comme ceux générés par les marques d'histones. Sur les données de γ-H2AX, MACS2 parvient à localiser le signal autour des coupures de l'enzyme (Fig. 32). Cependant, même en activant l'option pour appeler les pics larges, on obtient un grand nombre de postions enrichies autour de pics et non pas une zone large centrée sur une coupure comme on pourrait l'espérer.

Afin d'obtenir un résultat plus précis, j'ai décidé d'utiliser SICER, un logiciel spécialisé dans la recherche de zones enrichies larges (Xu et al. 2014). Autour des pics, le logiciel identifie des zones larges qui couvrent parfois plusieurs mégabases. Cela ne permet pas toujours de définir quelle cassure est à l'origine du signal, mais au lieu d'une centaine de positions autour d'une coupure, on obtient un bloc qui indique que le γ-H2AX est enrichi dans cette zone. Bien que ce résultat soit plus encourageant, le logiciel détecte beaucoup de faux positifs, correspondant à des zones sans site de coupure et aucun enrichissement apparent, y compris dans l'échantillon contrôle. Ce résultat est dû au fonctionnement du logiciel, qui détecte les zones concentrant plus de signal que le reste du génome, en fonction du signal total dans l'échantillon. Le logiciel donne un score FDR (False Discovery Rate) à chaque pic qu'il produit. Une valeur proche de zéro signifie que le logiciel considère la zone comme étant

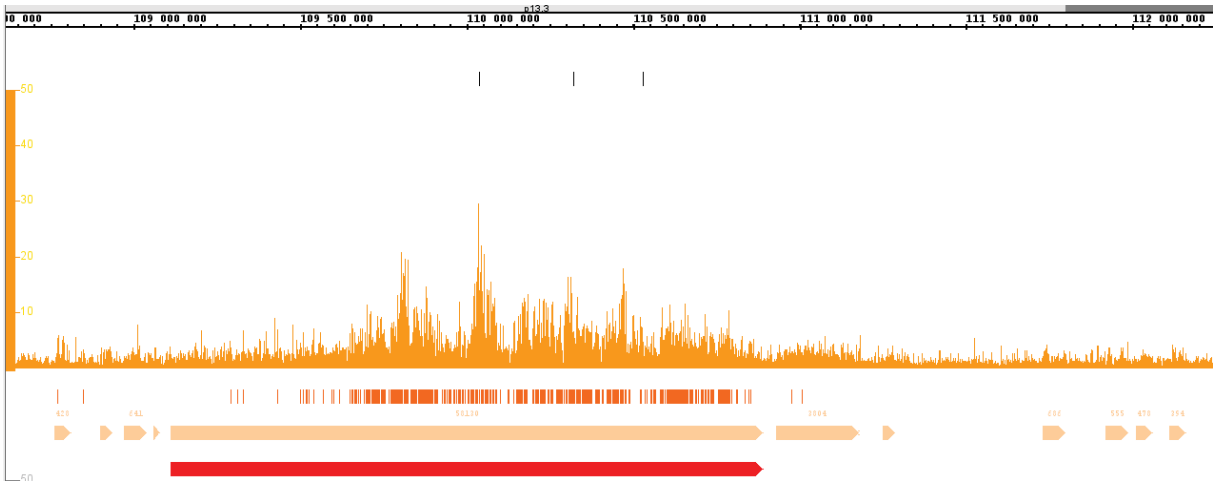


Figure 33 : Résultat du peak-calling sur le ChIP-Seq γ-H2AX des cellules U2OS. Le profil du signal est présenté en orange. De haut en bas, les résultats de MACS2 (orange foncé), SICER (orange clair) et du croisement des deux techniques (rouge).

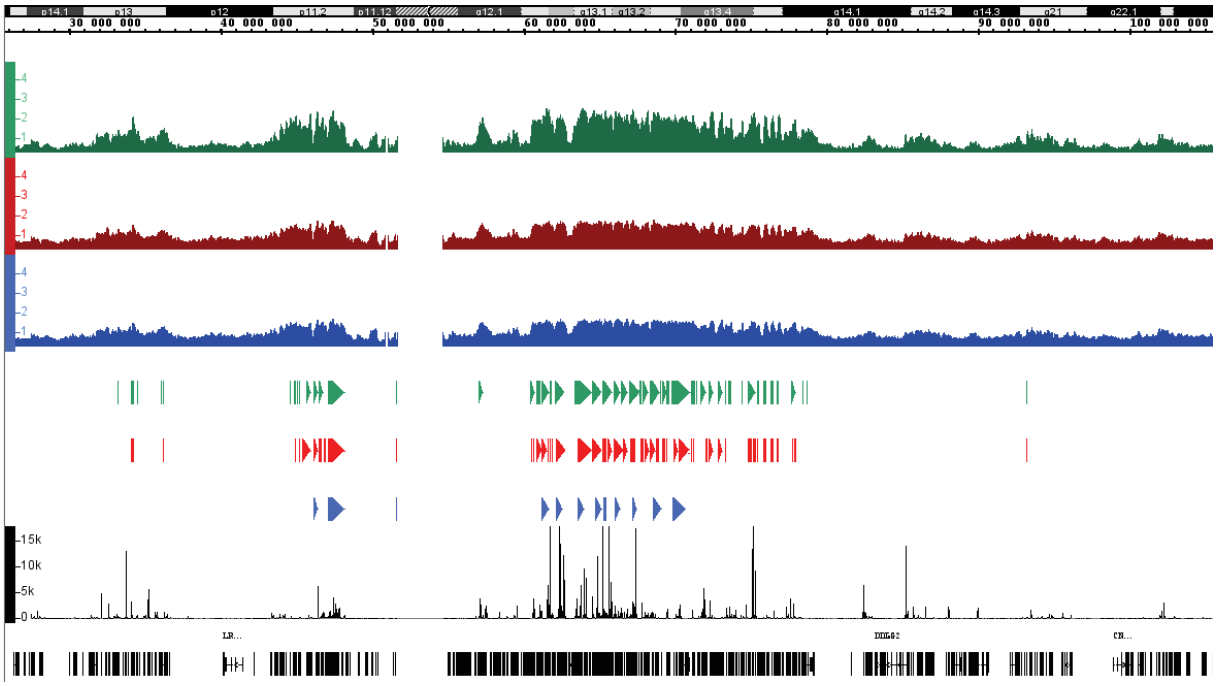


Figure 34 : Signal du γ-H2AX sur les cellules HeLa. Vu complète du chromosome 11. Le profil des trois lignées cellulaires est montré, ainsi que le résultat du peak calling. En bleu le contrôle, en rouge shASF et en vert shTOP1. En noir sont représenté les gènes et le profil RNA-Seq de nos données.

certainement enrichie. Même en filtrant sur cette valeur, un grand nombre faux positifs sont conservés, y compris dans le contrôle.

Grâce au logiciel de visualisation IGB (Fig. 33), on peut remarquer que les pics de SICER centrés sur coupures induites sont soutenus par un grand nombre de pics appelés par MACS2. Au contraire, les pics dus au bruit de fond font apparaître aucun signal lors du peak calling de MACS2. Afin d'obtenir un compromis entre la sensibilité de SICER et la spécificité de MACS2, j'ai sélectionné les zones définies par SICER ayant au moins cinq pics de MACS2 pour les supporter. On obtient par ce croisement des zones larges correspondant au signal étendu du γ-H2AX, tout en limitant les zones appelées aux positions autour des coupures induites.

4.1.2. Analyse des données de γ-H2AX dans les cellules HeLa

Les profils de γ-H2AX dans nos lignées cellulaires sont différents de ceux obtenus pour les cellules AsiSI-U2OS-ER. Notre laboratoire a montré par immunofluorescence que les cellules déplétées en TOP1 ou ASF/SF2 formaient plus de foyers de γ-H2AX. Notre hypothèse est que ce signal est dû à l'accumulation de R-loops et au stress réplicatif que cela engendre. Il est peu probable que les cassures soient aussi précisément localisées que dans les cellules AsiSI-U2OS-ER, contenant des sites de coupure très localisés et fréquemment coupés. Mais si une zone subit des cassures régulières, nous devrions obtenir une zone enrichie par rapport au bruit de fond.

Visuellement, la distribution du signal de γ-H2AX est très similaire dans nos trois conditions, mais on peut voir des différences d'intensité, particulièrement pour l'échantillon shTOP1 (Fig. 34). On ne détecte pas de fort enrichissement comme dans le modèle U2OS, mais des zones très larges, de plusieurs mégabases, où le signal s'accumule.

Après avoir appliqué le protocole précédemment décrit, j'ai identifié environ 200 zones enrichies dans la ligné contrôle, 1200 pour les cellules shASF et 1700 pour shTOP1. Les zones enrichies dans les cellules contrôle sont aussi détectées dans les cellules déplétées en ASF et TOP1. La majorité des zones appelées chez shASF le sont aussi chez shTOP1, mais l'inverse n'est pas vrai, seulement 50% des zones détectées dans la lignée shTOP1 le sont dans les cellules shASF.

Le γ-H2AX s'accumule particulièrement dans les zones riches en gènes. Dans l'échantillon shTOP1, plus de 90% des zones détectées couvrent une ou plusieurs annotations

Timing de réplication	S1	S2	S3	S4	S5	S6
Pourcentage de gènes γ-H2AX	70	34	21	12	9	9

Table 1 : Pourcentage des gènes marqués par le γ-H2AX, par timing de réplication. Résultats pour la ligné cellulaire shTOP1.

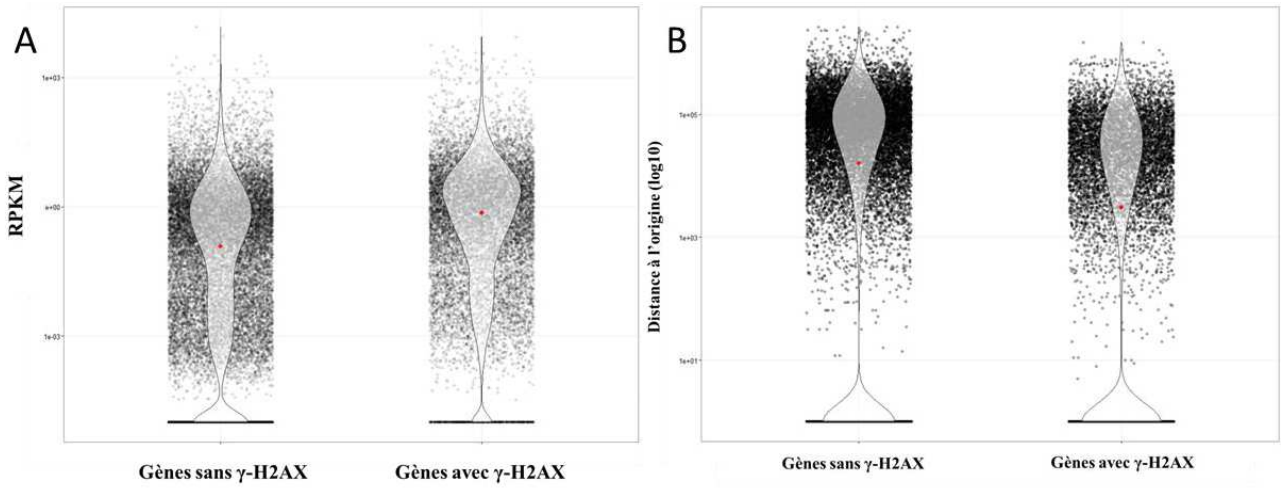


Figure 35 : Propriétés des gènes touchés par le γ-H2AX dans les cellules HeLa shTOP1 (Violin plot). (A) Niveau d'expression en RPKM des gènes marqués par le γ-H2AX est supérieurs au à celui des autres gènes dans les cellules shTOP1 (test de Wilcoxon : $p<2.2^{e-16}$). (B) Ensemble des annotations, classées en fonction de leur distance à l'origine de réplication forte la plus proche. Les gènes sont séparés en deux groupes, selon s'ils sont marqués par le γ-H2AX ou non dans les cellules shTOP1. Les gènes marqués sont en moyenne plus proches d'une origine forte (test de Wilcoxon : $p<2.2^{e-16}$). Le point rouge indique la médiane des valeurs.

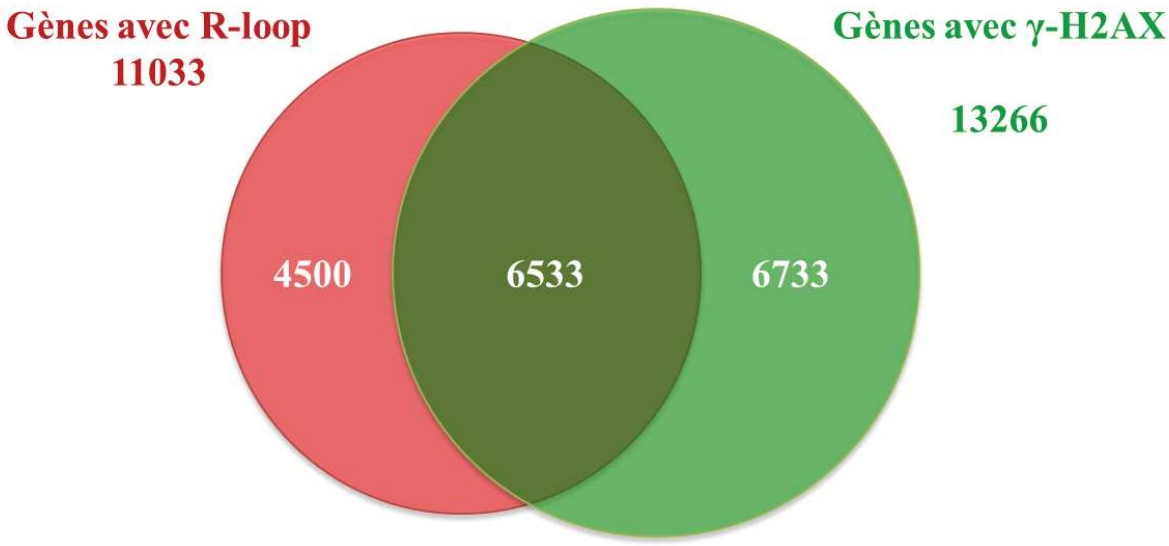


Figure 36 : Croisement des annotations marquées par le γ-H2AX et celles formant des R-loops dans la lignée cellulaire shTOP1.

(Refseq hg19). 37% des annotations sont marquées par du γ-H2AX, principalement des gènes codant pour des protéines et dont le niveau d'expression est en moyenne plus élevé (Fig 35). De plus, ces gènes sont répliqués tôt (Table 1) et plus proches des origines de réplication fortes (Fig. 35).

4.1.3. R-loops et γ-H2AX

Les gènes marqués par le γ-H2AX ont des caractéristiques similaires à celles trouvées pour les gènes formant des R-loops (cf. Publication) *i.e.* une forte expression, une réplication précoce au cours de la phase S et une proximité avec des origines de réplication majeures. Cependant, le croisement entre les deux marques n'est que partiel (Fig. 36), seulement la moitié des gènes marqués par le γ-H2AX forment des R-loops et un peu plus de la moitié des gènes formant des R-loops sont marqués par le γ-H2AX.

Les gènes marqués par le γ-H2AX appartiennent majoritairement à des clusters de gènes. Le signal de cassure pouvant s'étendre sur des mégabases, il est attendu que certains gènes couverts par le γ-H2AX ne sont pas à l'origine d'une cassure. Il est donc intéressant de noter que les gènes marqués ne formant pas de R-loops sont proches de gènes qui en forment (Fig. 37).

La relation entre R-loop et γ-H2AX n'est pas évidente au vu du nombre de gènes formant des R-loops et n'étant pas marqués (4500, Fig. 36). Il est connu que toutes les R-loops ne sont pas à l'origine de stress (Proudfoot, 2014), il est donc intéressant de décrire ce qui distingue ces deux groupes. Le niveau d'expression est légèrement plus élevé pour les gènes marqués par le γ-H2AX (Fig. 38), mais cette différence semble insuffisante pour expliquer l'origine du stress. En effet, à niveau d'expression équivalent, une partie seulement des gènes formant des R-loops sont positifs pour le γ-H2AX.

Notre hypothèse de travail stipulant que les R-loops bloquent l'avancée des fourches de réplication (Tuduri 2009), j'ai regardé si le timing de réplication est différent entre les gènes ayant une marque de stress et ceux qui n'en ont pas. La majorité des gènes formant des R-loops et ayant du signal de γ-H2AX sont répliqués très tôt (Fig. 39), à l'inverse les gènes répliqués tardivement et formant des R-loops ne sont pas sources de stress. Cependant, la distance aux origines fortes ne semble pas être la source du stress, de nombreux gènes marqués par le DRIP-Seq et ne formant pas de γ-H2AX ont une intersection avec une origine (Fig. 38).

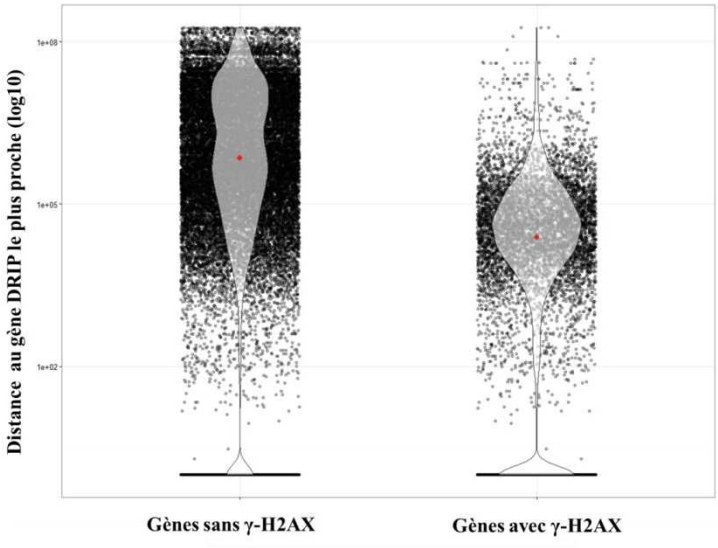


Figure 37 : Comparaison des distances en nucléotides entre les gènes ne formant pas de R-loop et le gène le plus proche en formant. Les distances sont calculées pour deux populations, les gènes marqués par le γ-H2AX et ceux qui ne le sont pas. Les résultats présentés correspondent aux cellules shTOP1.

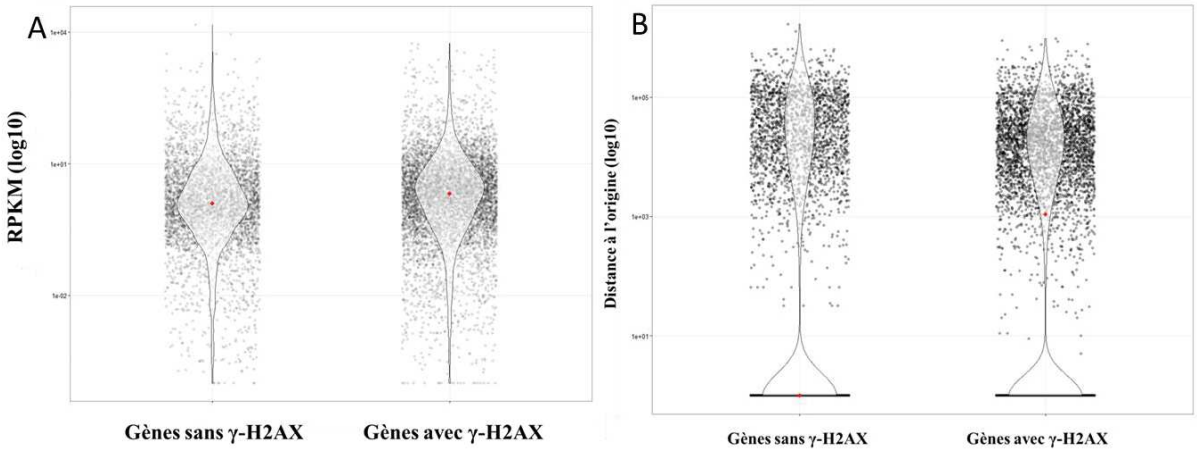


Figure 38 : Propriétés des gènes formant des R-loops et du γ-H2AX dans les cellules HeLa shTOP1 (Violin plot). Les résultats montrés sont similaires à ceux de la figure 35, mais ne concernent que les gènes formant des R-loops (A) Niveau d'expression en RPKM des gènes marqués par le γ-H2AX est supérieurs (test de Wilcoxon : $p<2.2^{e-16}$). (B) Distance à l'origine de réplication forte la plus proche. Les gènes sans γ-H2AX sont en moyenne plus proches (test de Wilcoxon : $p<2.2^{e-16}$). Le point rouge indique la médiane des valeurs.

5. Croisement du signal avec des données publiées (ENCODE)

Le projet ENCODE (Encyclopedia of DNA Elements, traduisible par Encyclopédie des éléments de l'ADN) est une collaboration internationale qui a pour objectif d'identifier tous les éléments fonctionnels du génome humain. Dans ce but un grand nombre de séquençages a été réalisé dans plusieurs lignées cellulaires, notamment les cellules HeLa. Grâce à ce projet, j'ai pu comparer la distribution de nos signaux à celles d'éléments de régulation tels que les facteurs de transcription, les marques d'histones et d'autres informations comme la distribution des polymérases.

5.1. Les marques d'histones

La méthylation et l'acétylation des histones dessinent le profil de la compaction de l'ADN. Au niveau des gènes, ces marques épigénétiques influencent tous les aspects de la transcription, le recrutement des polymérases, l'initiation, le niveau d'expression. Pour le projet ENCODE, les cartes génomiques d'une dizaine de ces marques ont été réalisées par ChIP-Seq. La majorité montre une activité de transcription, avec des variations comme une spécificité pour le promoteur (H3K9ac, H3K4me2, H3K27ac) ou pour les gènes fortement exprimés (H4K20me1). Il y a également une marque d'extinction de l'expression (H3K27me3).

Les profils que j'ai obtenus pour ces données indiquent un enrichissement des R-loops et du stress réplicatif sur les marques d'activation de la transcription (Fig. 40). Le signal le plus fort étant pour la marque H4K20me1. A l'opposé, le signal est inexistant au niveau de la marque d'extinction de la transcription.

Ces informations montrent le lien entre la transcription et la formation des R-loops. L'enrichissement en pRPA-S33 uniquement sur les marques d'activité de la transcription confirme que celle-ci est une source de stress pour la réplication. Lorsque l'on trie le signal sur l'intensité du DRIP-Seq, on observe la similarité de sa distribution avec celle du pRPA-S33. Il semble que dans toutes nos lignées cellulaires, aux positions marquant l'activité de la transcription, les R-loops sont associés à l'arrêt des fourches. Le signal de γ-H2AX est détecté uniquement pour la lignée shTOP1 et comme le pRPA-S33, il est enrichi aux positions où l'on détecte des R-loops. Ce signal est étalé et ne montre pas un enrichissement localisé comme les deux autres marques (Fig. 41).

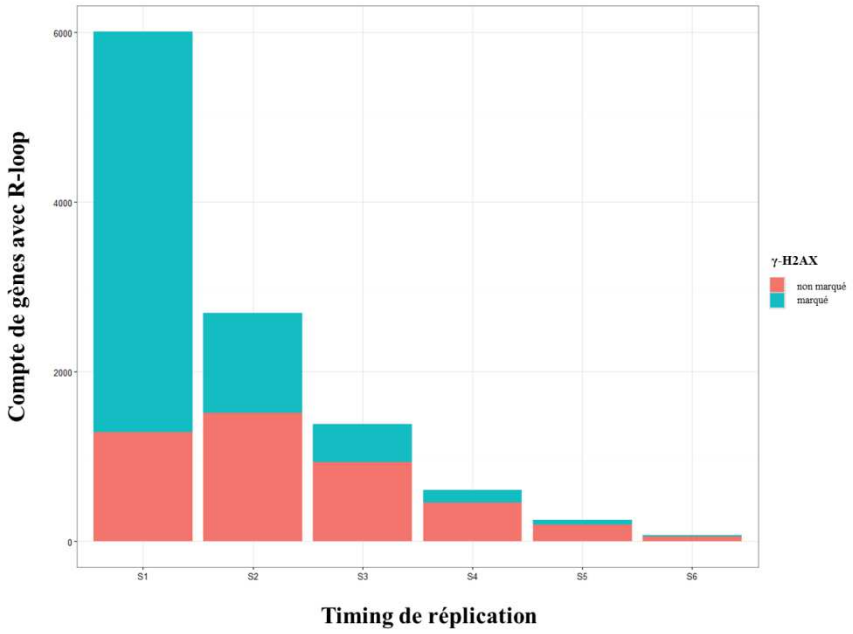


Figure 39 : Décompte des gènes formant des R-loops par phase de réplication. Les barres sont colorées en vert si les gènes sont couverts par du γ-H2AX et en rouge dans le cas contraire.

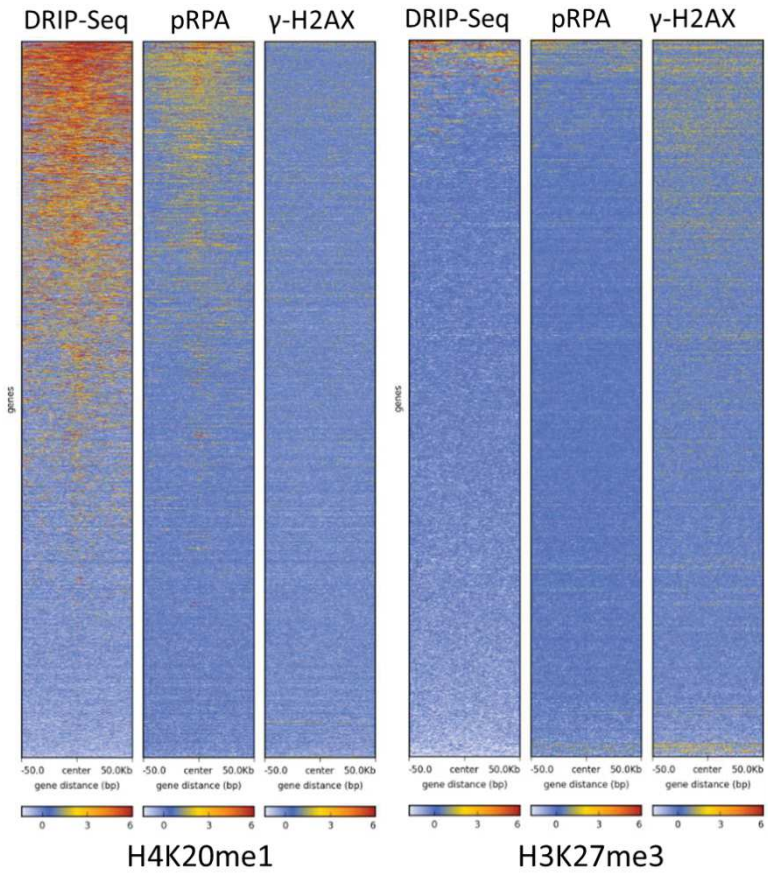


Figure 40 : Heatmaps des signaux de DRIP-Seq, ChIP-Seq pRPA et γ-H2AX pour les cellules shASF. Résultats représentés pour les positions sur le génome d'une marque d'activité de la transcription (H4K20me1) et d'une marque d'extinction de la transcription (H3K27me3). Les profils sont triés en fonction de l'intensité du DRIP-Seq.

5.2. Les ChIP-Seq de la polymérase II

Une autre donnée qui présente des profils intéressant pour tester notre hypothèse de départ est le ChIP-Seq de Pol2. Le projet ENCODE propose plusieurs résultats, réalisés avec différent anticorps, dans différents laboratoires à travers le monde et analysés avec différents logiciels de peak-calling. Dans l'ensemble, le peak-calling détecte les mêmes régions enrichies aux mêmes positions, malgré quelques variations dans la précision des pics appelés. La distribution du signal sur le génome est très similaire à celle qu'on obtient pour nos données de DRIP-Seq et pRPA-S33, avec un enrichissement encore plus fort aux extrémités.

La corrélation de notre signal avec les positions de la polymérase II est évidente (Fig. 42). Le signal est plus concentré que pour les positions d'histone, cela vient du fait que les pics appelés sont en moyenne plus fins. Il est intéressant de noter également que toutes les positions marquées par le DRIP-Seq ont un signal de pRPA-S33 et que, comme pour les histones, le signal de γ-H2AX est plus intense dans shTOP1 aux positions enrichies pour le DRIP-Seq.

Ces données lient directement l'activité de la polymérase II au stress réplicatif dans la cellule. L'absence de signal de pRPA-S33 aux positions qui ne sont pas enrichies dans le DRIP-Seq, suggère cependant que l'activité de la transcription ne suffit pas à bloquer le réplisome. Il faut la formation d'hybride par l'activité de la Pol2 pour générer le stress réplicatif. Cependant, l'enrichissement en γ-H2AX n'apparaissant que dans la souche shTOP1, il est possible que dans des conditions normales, une cellule soit capable de maintenir l'intégrité du génome à ces positions.

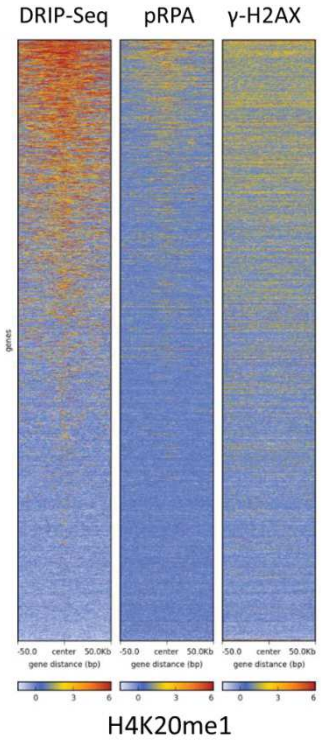


Figure 41: Heatmaps des signaux de DRIP-Seq, ChIP-Seq pRPA et γ-H2AX pour les cellules shTOP1. Les profils sont triés en fonction de l'intensité du DRIP-Seq et représenté pour une marque d'activité de la transcription (H4K20me1).

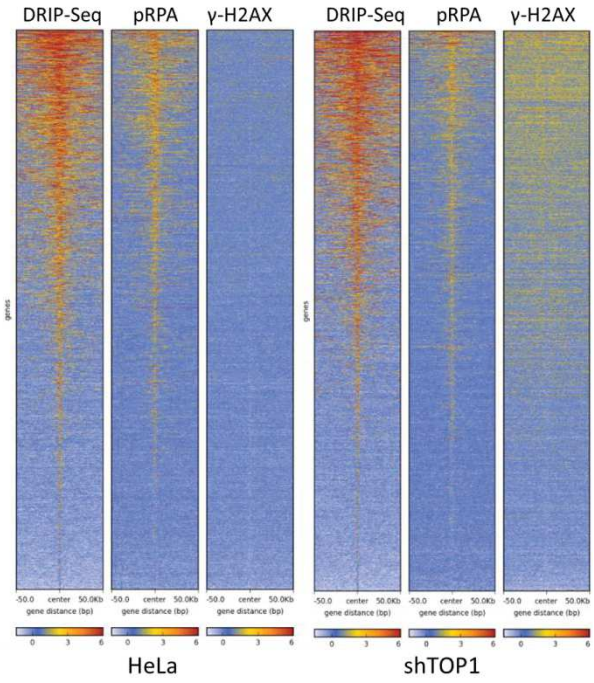


Figure 42: Heatmaps des signaux de DRIP-Seq, ChIP-Seq pRPA et γ-H2AX pour les cellules contrôle (HeLa) et shTOP1. Les profils sont triés en fonction de l'intensité du DRIP-Seq et représenté pour le positionnement de la polymérase 2 sur le génome (Données ENCODE).

Matériel et méthodes

1. Pipeline pour l'automatisation du traitement des données de séquençage.

Les séquenceurs de nouvelle génération produisent des millions de lectures. Obtenir des résultats analysables à partir de ces données brutes demande plusieurs étapes de traitement informatique (Cf Introduction : séquençage à haut débit). Etant donné la taille conséquente des données et la complexité de certaines étapes comme l'alignement sur le génome, leur préparation est longue et gourmande en ressources informatiques. D'autre part, il existe une multitude de méthodologies permettant leur analyse. Il est donc important d'avoir un système qui permet l'automatisation du traitement des données et la sauvegarde du protocole utilisé. C'est ce que permet la construction d'un pipeline, il connecte les différentes étapes sans intervention de l'utilisateur et le fichier de configuration utilisé permet de garder une trace des logiciels et des options choisis par l'utilisateur. Je vais décrire dans ce chapitre la méthode utilisée pour construire mon pipeline et les étapes qui le composent.

1.1. Le cluster genotoul

Le laboratoire ne disposant pas des ressources informatiques nécessaires au traitement des données de séquençage, j'ai demandé la création d'un compte sur la grappe de serveurs (cluster) Genotoul (http://bioinfo.genotoul.fr/). Genotoul est un Groupement d'Intérêt Scientifique (GIS) basé à Toulouse, qui offre des services dans le domaine de la science du vivant, à destination de l'académie et de l'industrie.

L'accès au cluster est gratuit avec 1 To d'espace mis à disposition pour les utilisateurs venant du milieu l'académique. Pour conserver nos données et réduire le nombre de transferts, nous avons également demandé 2 To de stockage sur des disques avec sauvegarde journalière.

Le cluster est composé d'une machine frontale pour la connexion et le stockage des données produites et de 68 nœuds disposant de 20 cœurs et 256 Go de mémoire vive. La grappe de serveurs fonctionne avec Sun Grid Engine (SGE). C'est ce système de gestion de grille informatique qui permet aux utilisateurs d'exécuter leur logiciel sur les nœuds de calcul.

1.2. Principe et fonctionnement du pipeline

Pour automatiser les traitements de nos données, j'ai développé une application en python, qui à partir d'un fichier de configuration, d'une liste de tâches à exécuter et d'un dossier d'entrées, va créer et lancer l'ensemble des étapes requises pour à la création des résultats.

Une fois exécuté, le pipeline vérifie que toutes les données nécessaires ont bien été fournies (fichier d'entrée, fichier de configuration, liste de tâches, dossier de sortie…). Le pipeline construit toujours le dossier de sortie selon le même plan, avec un dossier par type de résultat (i.e bam, peak-calling…), il est donc facile pour un utilisateur de retrouver l'information qui l'intéresse dans différents projets. L'étape suivante consiste en la création et l'exécution de scripts bash pour l'ensemble des tâches demandées. Quand tout est terminé, le pipeline envoie un e-mail à l'utilisateur si celui-ci l'a demandé.

Pour que le pipeline fonctionne, il faut une méthode permettant de lier les différentes étapes, de façon à ce que toute tâche en attente sache quand démarrer. Afin de connecter le lancement des différentes étapes de l'analyse, j'ai utilisé une propriété de SGE qui permet de nommer un job (une tâche envoyée sur le cluster) et d'utiliser cet identifiant pour demander à d'autres tâches d'attendre la fin de son déroulement pour se lancer. Le système reconnait le signe générique (*), qui remplace n'importe quelle suite de caractères dans les noms des jobs à attendre, j'ai donc pu créer des étapes qui nécessitent les résultats de plusieurs autres tâches. Par exemple, un script qui crée un rapport sur l'alignement de toutes les données attendra que l'ensemble des alignements soient terminés pour s'exécuter.

Le script bash contenant les commandes lancées, ainsi que les fichiers de sortie standard et de sortie d'erreur, est conservé avec les résultats. De cette façon, il est possible de revenir plus tard vérifier par exemple quelles options ont été utilisées pour un logiciel. Cela permet aussi de facilement relancer cette étape, simplement en exécutant le script.

Le choix des tâches à exécuter n'est pas contraint, un utilisateur peut choisir de lancer l'ensemble des logiciels (cf. paragraphe Les étapes du pipeline) ou au contraire d'arrêter ou de démarrer l'analyse à une étape intermédiaire. De cette façon, il est possible de changer certaines options d'une étape et d'accomplir uniquement celle-ci pour voir les changements produits.

1.3. Les scripts du pipeline

Le logiciel est composé d'un script principal (main.py, voir annexe 1) et d'un ensemble de scripts correspondant chacun à une tâche que l'on peut exécuter. Pour chaque étape, le principe de construction est le même, cela permet d'ajouter simplement de nouveaux logiciels en copiant un des scripts préexistants.

Le script principal prend en charge plusieurs processus. Il crée tout d'abord un identifiant unique de sept caractères. Ce code sera utilisé dans tous les noms de jobs créés, permettant ainsi de lancer plusieurs fois le pipeline sans craindre d'entremêlement des différents noms.

Le pipeline vérifie ensuite si les fichiers donnés en entrées correspondent aux tâches que l'utilisateur souhaite exécuter. Un certain nombre de logiciels nécessitent les résultats d'autres étapes, si le bon fichier d'entrée n'est pas fourni, le pipeline refuse de se lancer et indique à l'utilisateur les informations qui lui manquent.

Le script va ensuite récupérer la liste des noms des fichiers à traiter. Le système de nom est assez strict : il demande des extensions et parfois un code précis. Par exemple, les fichiers .fastq fournis, doivent être au format zippé (.gz) et si le séquençage est paired-end, leur nom doit être identique et se finir par _1.fastq.gz et _2.fastq.gz pour identifier les paires.

Enfin le pipeline va appeler chaque script correspondant à une étape demandée. Il fournit à chaque fois un dictionnaire[1], contenant les options pour l'ensemble des logiciels, obtenues grâce au fichier de configuration, ainsi que la liste de noms correspondant aux fichiers d'entrées. Chaque appel à un autre script est construit de la même façon, il vérifie que la tâche est dans la liste demandée par l'utilisateur, puis si les étapes précédentes nécessaires à son exécution ont bien été appelées. Si tout est en ordre, alors il lance le script.

Pour la même raison, les scripts appelés sont eux aussi construits selon un modèle unique. Ils contiennent deux fonctions. Une fonction qui permet au script principal de les appeler et une fonction qui crée et exécute les scripts bash pour chaque échantillon. Pour cela, ils utilisent la liste de noms fournis par le script principal qui fait partie du dictionnaire contenant également toutes les options choisies par l'utilisateur.

1.4. Utilisation du pipeline

Pour l'utilisateur, la partie la plus importante du pipeline est le fichier de configuration. Il s'agit d'un fichier au format CSV (Comma-Separated Values), qui réunit toutes les informations essentielles à l'exécution des étapes du pipeline (cf. exemple de fichier de configuration en annexe). Une partie du fichier contient des informations stables, qui ne varient quasiment jamais, comme le chemin vers les logiciels utilisés ou l'adresse mail de l'utilisateur. L'autre partie concerne le choix les logiciels à utiliser et les options que

[1] Un dictionnaire est une table spéciale en python. Il fonctionne comme une liste, mais au lieu de stocker les informations dans un ordre précis, il associe chaque objet contenu à une clé. Par exemple, un nom associé à un numéro dans un carnet d'adresse. Si on lui donne un nom le dictionnaire retourne le numéro correspondant.

l'utilisateur veut leur donner. Par exemple, pour le peak-calling, il est possible d'utiliser MACS2 ou SICER, puis de donner les options que l'utilisateur choisit. Ce fichier peut ensuite être utilisé pour reproduire le processus sur de nouvelles données ou partagé avec d'autres utilisateurs qui sont alors sûrs d'exécuter la même analyse.

Une fois que ce fichier est créé, il suffit d'appeler le pipeline en lui donnant le fichier de configuration, le dossier où trouver les données d'entrées, celui où écrire les résultats et la liste des tâches à exécuter (Fig. 43).

```
python main.py –fastq /path/myFASTQdir –od outDir –cf config.txt –t 1 2 3 8
```

Figure 43 : Exemple de commande shell pour exécuter le pipeline. Dans cet exemple, l'utilisateur donne un dossier (`myFASTQdir`) avec des fichiers .fastq.gz en entré et indique au logiciel au le trouver avec l'option « -fastq ». Il donne ensuite un dossier où écrire les résultats avec l'option « -od » (pour Output Directory). Il indique ensuite où trouver le fichier de configuration « -cf ». Enfin il précise les taches qu'il veut exécuter « -t » (pour Task). La signification des numéros de tâches est décrite dans le texte principal.

1.5. Les étapes du pipeline

L'objectif du logiciel est d'exécuter en une seule fois les tâches les plus récurrentes du traitement de données. Certaines étapes ne sont pas utiles à tous les projets, par exemple de peak calling pour des données RNA-Seq, c'est pourquoi l'utilisateur est libre de donner la liste de tâches qu'il souhaite (Fig. 44).

Voici la liste des étapes actuellement disponibles, avec les logiciels utilisés pour les réaliser :

1. Contrôle qualité du séquençage : FastQC et un script PERL qui crée un rapport sur la qualité, le nombre de lectures, le nombre de lectures uniques, la séquence et le compte de lecture les plus répétés.
2. Alignement des lectures sur le génome de référence : Bowtie2 ou STAR (Langmead et al. 2012, Dobin et al. 2013)
3. Contrôle qualité de l'alignement : SAMtools flagstat (Li et al. 2009) et un script python pour vérifier le nombre de lectures alignées, la qualité de l'alignement, le nombre de positions uniques du génome où une lecture s'est alignée, la proportion

que cela représente de l'alignement, la position avec le plus grand nombres d'alignements et la proportion que cela représente.

4. Création de profil .bigwig pour la visualisation : DeepTools (Ramírez et al. 2012)
5. Création de fichiers .bed pour le logiciel SICER : bedTools bamtobed (Quinlan 2002)
6. Peak-calling : MACS2 ou SICER (Zhang et al. 2008, Xu et al. 2014)
7. Ratio au format bigwig, pour la visualisation : Deeptools ne fonctionne que si un fichier, contenant les paires d'échantillons à utiliser pour les ratios, est donné en entrée. Ce fichier est une simple liste au format TSV (Tab-Separated Value) avec deux colonnes de noms de fichiers, le ratio étant fait pour chaque ligne entre l'échantillon de la première colonne et celui de la seconde colonne.
8. Compte de lecture par annotation ou RPKM par annotation : bedTools multicov. L'utilisateur doit donner un dossier en entrée qui contienne au moins une annotation au format .bed, .gff ou .vcf. Si le dossier contient plusieurs annotations, les comptes seront faits pour chaque échantillon sur toutes les annotations.
9. Correlation et PCA: Deeptools, multiBamSummary, plotCorrelation et plotPCA.

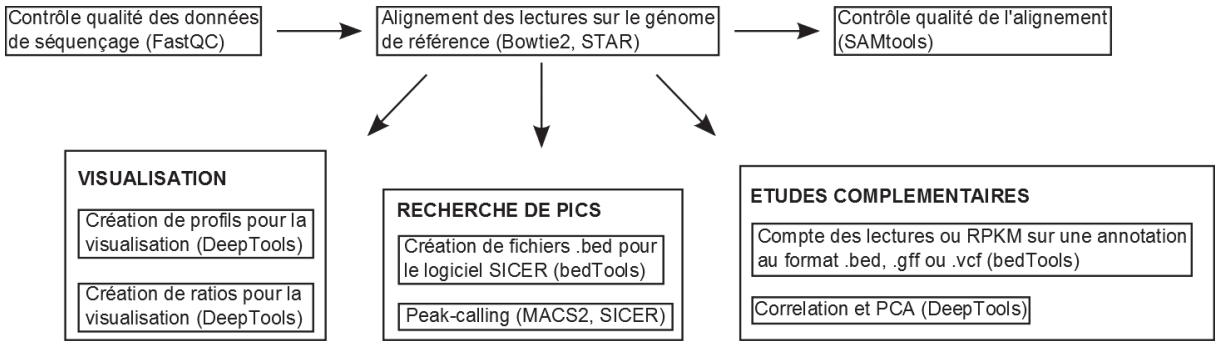


Figure 44 : Étapes du pipeline. Les étapes sont réunis par intérêt et les flêches représente les dépendances à l'interieur du pipeline.

1.6. Discussion

Ce pipeline a servi à préparer toutes les données du laboratoire depuis quatre ans. Grâce à l'automatisation du traitement, les résultats d'un séquençage sont prêts à être analysés les jours suivant leur mise à disposition par nos collaborateurs. Sa simplicité d'utilisation a permis à certains biologistes de prendre la main sur le traitement des données et certaines étapes ont été ajoutées à leur demande.

Plusieurs outils ont été développés ces dernières années pour permettre de construire des pipelines comme celui-ci. Ces solutions offrent plus de souplesse dans les mises à jour et encouragent le partage de code dans une communauté de plus en plus large d'informaticiens et de bioinformaticiens. C'est le cas de Snakemake (Köster et al. 2012), qui se présente sous la forme d'un langage de programmation, basé sur python, spécifiquement dédié à la création de pipeline. Il est particulièrement intéressant car il fonctionne avec un système de blocs de code, appelés règles, qu'il est possible d'intégrer ensuite dans d'autres pipelines mais également de partager avec la communauté. Ainsi il n'y a plus besoin de recréer une règle pour le logiciel Bowtie si celle-ci est disponible dans le répertoire de règles en ligne. De plus, il s'adapte aux différents systèmes de gestion de queue existant et peut même être lancé sur un serveur simple.

Discussion

Le stress réplicatif est une caractéristique récurrente des cellules tumorales qui apparaît dès les premières étapes de la maladie. Un modèle a été proposé, dans lequel l'activation des oncogènes, qui entraîne une prolifération aberrante des cellules, provoque le stress réplicatif, les cassures de l'ADN et l'instabilité génomique (Halazonetis et al. 2008). Bien que ce modèle explique le lien entre activation des oncogènes et l'accumulation de mutations, les mécanismes qui lient l'accélération de la prolifération et le stress réplicatif restent à éclaircir.

Il est maintenant bien établi que la transcription contribue au stress réplicatif (Bermejo et al. 2012, Jones et al. 2013, Helmrich et al. 2013) et particulièrement via la formation de R-loops (Huertas et Aguilera 2003, Tuduri et al. 2009). La séparation spatiale et temporelle de la réplication et de la transcription limite normalement ces conflits, mais l'organisation fonctionnelle du génome étant fortement perturbée dans de nombreux cancers (Madakashira et Sadler 2017), la transcription pourrait être la principale source de stress réplicatif dans les cellules cancéreuses (Hamperl et Cimprich 2016).

Afin de mieux comprendre le rôle des R-loops dans ces conflits, nous avons développé des modèles cellulaires permettant d'accroitre la formation des R-loops. Nous avons notamment choisi d'induire une déplétion de TOP1, qui supprime le stress topologique en amont et aval des polymérases, et d'ASF/SF2, qui prend en charge l'ARN naissant. La déplétion de ces protéines augmente la formation des R-loops, ralentit la progression des fourches de réplication et entraine l'instabilité du génome (Li et Manley 2005, Tuduri et al. 2009). Dans ces cellules, la réplication peut être rétablie par la surexpression de RNase H1, ce qui suggère l'existence d'un lien entre la formation des R-loops et le stress réplicatif. Il reste à déterminer si les R-loops impactent directement la réplication en bloquant l'avancée des fourches et quelles conditions provoquent leur toxicité.

Les R-loops participent à de nombreux processus physiologiques, notamment l'initiation et la terminaison de la transcription (Ginno et al. 2012, Skourti-Stathaki et al. 2014, Lang et al. 2017). Pour étudier leur effet sur la réplication, nous avons utilisé deux lignées cellulaires dérivées de cellules HeLa S3, dans lesquelles l'expression inductible d'un shRNA entraine la déplétion de TOP1 ou d'ASF/SF2. Grâce aux techniques de séquençage à haut débit, nous avons pu établir une carte des zones du génome favorables à la formation des R-loops (DRIP-Seq). Afin d'observer le lien possible entre R-loop et stress réplicatif, nous avons également réalisé des expériences de ChIP-Seq pour positionner l'arrêt des fourches de réplication et les

cassures de l'ADN, avec des anticorps dirigés contre γ-H2AX et contre une forme de RPA phosphorylée par ATR.

1. Le DRIP-Seq

L'accumulation de R-loops dans nos lignées cellulaires déplétées pour TOP1 ou ASF/SF2 a d'abord été confirmée par immunofluorescence (IF) et par slot blot. La technique de DRIP-Seq, qui associe l'immunoprécipitation des hybrides ADN/ARN grâce à l'anticorps S9.6 au séquençage à haut débit, nous a ensuite permis de réaliser une carte génomique de la position des R-loops dans les cellules HeLa. Nous avons obtenu des résultats de bonne qualité, comparables entre réplicas et proche du signal obtenu dans la lignée cellulaire Nterra2 de carcinome embryonnaire (Sanz et al. 2016).

La distribution des R-loops sur le génome dans les cellules contrôles et dans les cellules déplétées pour ASF/SF2 ou TOP1 est très similaire. La majorité du signal est partagée entre les trois lignées cellulaires, cependant l'analyse par peak calling confirme la formation de nouvelles R-loops dans nos cellules déplétées.

Les résultats montrent que la formation des R-loops est principalement liée à la transcription. En effet, 85% du signal est associé à des annotations de gènes (RefSeq). Les sites d'initiation et de terminaison sont particulièrement enrichis avec respectivement 10% et 15% du signal détecté. Ces résultats sont en accord avec des études précédentes qui décrivent le rôle des R-loops à l'initiation et à la terminaison des gènes (Skourti-Stathaki et al. 2011, Ginno et al. 2012). La distribution des R-loops sur les gènes (corps du gène, TSS et TTS) est identique dans les cellules déplétées pour ASF/SF2 ou TOP1 et le contrôle. Cela signifie que même dans des conditions favorisant leur apparition, les R-loops se forment préférablement sur le corps des gènes et n'apparaissent pas spontanément à d'autres positions du génome.

2. Les marques de stress réplicatif

Pour étudier le stress réplicatif dans nos cellules et l'instabilité génomique qui en découle, nous avons décidé de réaliser des ChIP-Seq dirigés contre pRPA-S33 et de γ-H2AX, qui marquent respectivement l'arrêt des fourches de réplication et les cassures de l'ADN.

Notre objectif étant de caractériser l'impact des R-loops sur la réplication dans des conditions normales de croissance, nous avons réalisé nos expériences en absence de stress génotoxique d'origine exogène, ce qui complique significativement le traitement des données.

En effet, l'analyse de données de ChIP-Seq passe le plus souvent par une recherche de pics ou de zones enrichies. Cela fonctionne très bien si le signal étudié est fort et concentré en des positions précises du génome, comme c'est le cas pour les facteurs de transcription. L'analyse est plus complexe pour des signaux plus étendus, à l'image des marques d'histones, mais les logiciels parviennent tout de même à produire de bons résultats. Le défi de nos données est que le signal n'est pas attendu à une position précise et qu'il ne concerne probablement qu'une partie de la population de cellules pour une position donnée. Nous nous attendions donc à un signal faible, potentiellement enrichi au niveau des régions du génome difficiles à répliquer.

2.1. Le signal de pRPA

Le signal que nous avons obtenu pour le pRPA-S33 est effectivement faible, mais l'analyse avec le logiciel MACS2 nous a permis d'extraire certaines positions enrichies. Les pics obtenus corrèlent fortement avec ceux du DRIP-Seq et les résultats montrent un très fort enrichissement du signal sur les gènes, et plus particulièrement au TTS. Dans nos trois lignées cellulaires, un peu plus de 80% du signal est proche ou sur un transcrit et environ 90% des gènes marqués par le pRPA sont également enrichis en R-loops.

Etonamment, nous avons identifié plus de pics de pRPA-S33 dans les cellules contrôle que dans nos lignées déplétées en TOP1 et ASF/SF2, ce qui est surprenant car les données d'immunofluorescence, de western blot et de DNA fiber spreading montrent que le stress réplicatif est plus intense dans les cellules déplétées pour ASF/SF2 ou TOP1. Une explication possible est qu'en l'absence de stress, l'arrêt des fourches fait partie d'un processus physiologique, avec une position plus restreinte sur le génome et un comportement identique dans l'ensemble de la population. Le signal est alors plus localisé, plus intense et le logiciel peut plus facilement le distinguer du bruit de fond. Alternativement, il est important de souligner que le nombre de pics n'est pas nécessairement le reflet de l'intensité globale du signal, un pic majeur dans une lignée pouvant être fragmenté en une multitude de pics mineurs dans une autre lignée.

La recherche de pics que j'ai effectuée était très conservative, pour ne récupérer que les pics les plus intenses et pour lesquels le logiciel donnait un bon score de confiance. N'ayant pas de répliques biologiques pour ces données, j'ai décidé de les exploiter principalement à travers des représentations graphiques, telles que des heatmaps et des profils moyens. Ces

résultats visuels confirment la corrélation entre R-loops et pRPA et montrent que, comme pour le peak calling, le plus grand nombre d'arrêts de fourche est associé aux R loops.

2.2. Le signal de γ-H2AX

La phosphorylation du variant d'histone H2AX (γ-H2AX) par ATM, ATR et DNA-PK peut s'étendre sur plusieurs millions de bases à partir d'une cassure de l'ADN. Notre utilisation comme contrôle positif du modèle cellulaire développé par l'équipe du Dr Legube (Iacovoni et al. 2010), qui permet d'induire des dizaines de cassures dans le génome, montre que le signal de γ-H2AX est déjà difficile à extraire, même quand la position des cassures est préalablement connue. Les zones de cassures sont fortement enrichies pour le signal de γ-H2AX, ce qui montre que l'immunoprécipitation de cette marque fonctionne bien pour nos données. Cependant, le signal est très étalé et il est très fragmenté. Selon le logiciel utilisé, on peut obtenir de nombreux pics mineurs dans la région qui entoure une coupure (MACS2) ou de grandes zones couvrant de quelques dizaines de kilobases à plusieurs mégabases (SICER). Le logiciel SICER obtenant un résultat plus proche de la réalité du signal mais montrant une faible spécificité, j'ai décidé de croiser les résultats des deux logiciels pour obtenir un profil correspondant à l'enrichissement observé. Malgré cela, lorsque plusieurs sites de restriction sont proches sur le génome, il est impossible de distinguer si le signal est provoqué par une seule ou par plusieurs coupures enzymatiques.

Le signal obtenu pour nos données reflète toutes ces difficultés. Les zones enrichies appelées par les logiciels sont extrêmement étendues, et leur signal couvre le plus souvent de nombreux gènes. Ce phénomène est d'autant plus fort que les parties du génome où l'on détecte le plus de signal sont des clusters de gènes dans lesquels bien souvent les transcrits se chevauchent ou ne sont qu'à quelques kilobases les uns des autres.

Les zones enrichies sont pour la plupart partagées entre nos lignées cellulaires. On distingue cependant un enrichissement plus fort dans nos cellules déplétées pour TOP1. Ce résultat indique que le stress provoqué par les R-loops dans la lignée shASF/SF2 peut être suffisant pour arrêter les fourches, mais qu'il faut l'ajout une contrainte topologique pour entrainer l'instabilité génomique, comme dans la lignée shTOP1.

3. Descriptions des zones favorables aux R-loops et au stress réplicatif

L'analyse des résultats du DRIP-Seq et des expériences de ChIP-Seq dans nos lignées cellulaires a montré que les zones riches en gènes étaient les plus instables. Cependant tous

les transcrits ne forment pas de R-loops. De même, l'immense majorité des gènes enrichis en R-loops ne sont pas associés à du stress réplicatif. Pour identifier les conditions qui favorisent la formation des hybrides et ce qui fait qu'une partie d'entre eux représente un danger pour la cellule, j'ai utilisé d'autres données de séquençage, parfois réalisées sur nos cellules (RNA-Seq, timing de réplication par micro-array) et parfois publiées par d'autres laboratoires travaillant avec les cellules HeLa (OK-Seq, GRO-Seq).

3.1. Le niveau d'expression des gènes

Pour pouvoir analyser l'expression des gènes dans nos cellules, nous avons réalisé un RNA-Seq pour chacune de nos lignées cellulaires. Nous avons constaté que les variations du niveau de transcription entre la lignée contrôle et les lignées déplétées sont minimes. En effet, seulement une centaine de gènes montrent un changement dans leur niveau d'expression. Ces gènes ne forment pas de groupes cohérents, ils ont différentes tailles et niveaux d'expression, ne sont pas regroupés dans le génome et l'analyse ontologique montre qu'aucune catégorie particulière n'est enrichie dans cet ensemble de gènes.

J'ai utilisé ces données pour étudier l'influence de la transcription sur la formation des R loops. Les résultats indiquent que celles-ci sont enrichies dans les gènes actifs et dont le niveau d'expression est élevé. Au contraire, très peu de gènes faiblement transcrits forment des R-loops. Dans les lignées cellulaires déplétées, les nouvelles R-loops se forment en partie sur certains gènes fortement exprimés, mais le différentiel d'enrichissement est surtout détecté au niveau des gènes ayant un niveau de transcription moyen. J'ai émis plusieurs hypothèses pour expliquer le fait que des R-loops soient détectables sur ces nouveaux gènes dans nos lignées déplétées pour ASF/SF2 ou TOP1, la plus simple étant qu'ils ne forment pas de R-loops en conditions normales mais que la déplétion d'ASF/SF2 ou de TOP1 favorise la formation d'hybrides pour les raisons mentionnées plus haut. Il est possible que ces gènes produisent des R-loops dans les cellules normales, mais que le niveau de signal ne permette pas leur détection. En effet, une R-loops ayant une demi-vie de 10 minutes (Sanz et al. 2016), si les R-loops se forment au cours de la transcription, alors nos chances de les détecter varient avec le niveau d'expression des gènes. Il est donc possible que les nouvelles R-loops détectées se forment aussi dans une cellule normale, mais qu'il faille une condition comme l'absence d'ASF/SF2 pour les stabiliser et les détecter par DRIP-Seq.

D'autres gènes ont un niveau d'expression élevé mais ne forment pas de R-loops dans les cellules Hela contrôles. La formation des R-loops, bien qu'elle ne soit pas spécifique à la

séquence des gènes, est fortement influencée par la composition de l'ADN, un GC% élevé etun déséquilibre dans le GC skew favorisant leur formation (Ginno et al. 2013). Dans ces gènes, l'absence de TOP1 ou ASF/SF2 pourrait favoriser la génération d'hybrides sur des séquences qui n'en forment pas normalement.

Ces résultats suggèrent que les R-loops ne se forment pas à chaque passage de l'ARN polymérase, mais il y a une probabilité que cette structure apparaisse en fonction de la composition de la séquence ADN. En conséquence, pour un même niveau d'expression, en fonction de leurs séquences, deux gènes ne produiront pas la même quantité de R-loops.

L'arrêt des fourches détecté par pRPA-S33 montre le même enrichissement pour les trois lignées au niveau des gènes fortement exprimés. Ce résultat n'est pas surprenant, car 90% du signal corrèle avec celui du DRIP-Seq. Cependant, le biais vers les transcrits de niveau d'expression moyen que l'on observe par DRIP-Seq dans nos cellules déplétées n'apparaît pas pour le signal de pRPA-S33. Il est également important de noter que l'arrêt des fourches est principalement détecté sur les gènes formant déjà des R-loops dans les cellules HeLa. Dans la lignée shASF seulement 2% des nouveaux gènes sont marqués par le pRPA-S33, contre 18% des gènes formant déjà des R-loops dans le contrôle. Un résultat qui est confirmé par les heatmaps, qui montrent un signal beaucoup plus faible de pRPA-S33 et de R-loops pour les nouveaux gènes formant des hybrides.

Ces résultats indiquent que la formation d'hybrides sur de nouveaux transcrits n'est pas la source principale d'arrêt des fourches dans nos cellules. Il semble que l'accumulation de R-loops et la stabilisation de ces dernières, aux positions qui en forment déjà dans les cellules contrôle, soient plus toxiques pour la cellule. Cela suggère également que la formation d'une R-loop n'est pas suffisante pour induire un arrêt de fourche et que d'autres facteurs déterminent la capacité d'une R-loop donnée à interférer avec la réplication.

3.2. Le timing de réplication et l'orientation des fourches

Les fourches de réplication étant ralenties en absence de TOP1 ou ASF/SF2 (Tuduri et al. 2009), nous avons comparé le timing de réplication dans nos cellules déplétées avec celui des cellules contrôles. Pour cela, en collaboration avec Jean-Charles Cadoret (Institut Jacques Monod, Paris), nous avons réalisé une analyse de la réplication par microarray. Le résultat a montré peu de différences entre le timing de nos cellules contrôles et celui des cellules déplétées, même si certaines régions ont montré des transitions de précoce vers tardif ou de

tardif vers précoce. De plus, nous avons constaté que ces zones de transition ne sont pas particulièrement enrichies en R-loops, ce qui suggère que le changement de timing est une conséquence indirecte du stress réplicatif induit par les R-loops. Enfin, en comparant nos résultats avec les données publiées de Repli-Seq publié pour les cellules HeLa (Hansen et al. 2010), j'ai pu constater que le timing était très similaire. Le ralentissement des fourches dans nos cellules déplétées pour TOP1 ou ASF/SF2 ne semble donc pas fortement perturber l'organisation générale du timing de réplication.

Les zones riches en gènes sont répliquées précocement (Koren et al. 2012). Cependant, ce biais de répartition est plus fort pour les gènes formant des R-loops. En divisant la phase S en 6 sections (S1 à S6), nos résultats indiquent que 80% des gènes formant des R-loops sont répliqués en début de phase S (S1, S2), contre 60% pour l'ensemble des gènes. De la même façon, les gènes ayant un niveau d'expression élevé sont préférentiellement répliqués précocement. Il est donc logique, au vu de la corrélation entre R-loops et transcription, que la majorité des gènes formant des hybrides soient répliqués en début de phase S. De plus, les gènes ayant un niveau de transcription moyen et faible, pour lesquels on détecte la formation de R-loops, sont aussi répliqués en début de phase S. Ce résultat renforce l'idée que les gènes répliqués précocement ont plus de chances de former des R-loops, quel que soit leur niveau d'expression.

Pour ce qui est du timing de réplication, nous avons constaté que la distribution du signal de pRPA-S33 montre le même biais par rapport au timing de réplication que le DRIP-Seq. Ce biais est conservé si l'on observe la sous-population des nouveaux gènes formant des R-loops dans nos lignées shASF et shTOP1.

D'autre part, la proximité d'une origine de réplication et l'orientation avec laquelle la fourche rencontre la transcription pourrait avoir une influence sur la formation des R-loops et du stress réplicatif. Récemment, l'équipe du Dr Cimprich a montré que l'orientation de la rencontre, entre les ARN polymérases et les ADN polymérases, influence la formation des R-loops (Hamperl et al. 2017). Selon ces résultats, les collisions frontales (Head-On, HO) favorisent la formation d'hybrides alors que les rencontres co-directionnelles (CD) diminuent la quantité de R-loops.

Pour étudier l'influence de la proximité d'une origine et l'orientation des fourches de réplication sur la formation des R-loops, j'ai utilisé les données de OK-Seq produites dans les cellules HeLa par les équipes d'Olivier Hyrien et Chun-Long Chen (Petryk et al. 2016). Le

OK-Seq permet de définir des zones d'initiation de la réplication allant de quelque kilobases à une centaine de kilobases. Il y a donc de nombreux cas où une ou plusieurs origines de réplication sont contenues dans le corps d'un gène. Inversement, certaines zones d'initiation contiennent plusieurs gènes. Dans cette sous-population, comme dans le reste du génome, la majorité des transcrits ne forment pas de R-loops, mais on constate tout de même que la proportion de gènes formant des hybrides y est plus importante.

Si l'on considère la population de gènes qui ne contiennent pas d'origines de réplication, on observe que les gènes formant des R-loops sont, en moyenne, plus proches d'une origine forte. Néanmoins, bien que les collisions frontales favorisent la formation de R-loops, on ne trouve pas de déséquilibre dans la proportion d'origines en amont ou en aval de ces gènes. Les origines que j'ai utilisées pour cette analyse ne représentent pas l'ensemble des origines actives sur le génome, mais simplement une sous-population d'origines dont le signal est plus fort et plus constant. Il est donc difficile de conclure que l'orientation n'a pas d'effet sur la formation de R-loops, car il est possible que d'autres origines soient en charge de la réplication des transcrits étudiés.

Une autre façon d'aborder le problème est de regarder la transcription autour des origines fortes et de considérer l'orientation HO et CD des gènes environnants. Dans ce but, l'équipe du Dr Cimprich a réalisé une expérience qui illustre l'effet de l'orientation des rencontres entre polymérases sur la formation des R-loops. Ils ont pour cela sélectionné un ensemble d'origines se trouvant sur le corps de gènes et pour lesquelles il y a suffisamment de distance entre l'origine est les extrémités des gènes, pour que la formation des R-loops à ces positions n'influence pas leurs résultats. Ils ont ensuite calculé le profil moyen du signal de DRIP-Seq autour de ces origines. Le résultat confirme l'influence de l'orientation de la fourche par rapport à la transcription sur la formation de R-loops. L'orientation HO montre une augmentation la formation des R-loops et l'orientation CD une diminution. En utilisant les origines qu'ils ont sélectionnées, j'ai pu reproduire leurs résultats avec nos données de DRIP-Seq. J'ai retrouvé l'effet des rencontres sur la formation des R-loops dans nos trois lignées cellulaires. Cependant, je n'ai pas observé d'effet sur le signal de pRPA-S33, ce qui indique que les rencontres de la réplication et de la transcription autour de ces origines intergéniques ne provoquent pas l'arrêt des fourches.

L'accumulation de stress topologique et le ralentissement des polymérases favorisent la formation de R-loops. Dans cette situation, l'ADN est ouvert à cause des supertours accumulés et l'ARN naissant reste plus longtemps à proximité de son brin complémentaire. Cela peut expliquer le fait qu'une rencontre de front ait plus de chances d'entraîner la formation de R-loops. Cependant le profil de pRPA-S33 ne semble pas indiquer de stress topologique fort au niveau de ces origines. Ce résultat est intéressant, mais il pourrait être biaisé par le choix de ces origines intergéniques. Pour étendre cette étude aux origines intergéniques, qui sont les plus abondantes, j'ai utilisé des données de GRO-Seq générées dans des cellules HeLa (Andersson et al. 2014) pour définir l'orientation de la transcription dans des fenêtres de 15 kb de chaque côté des origines de réplication identifiées par OK-Seq. J'ai ainsi pu définir deux groupes de régions orientées majoritairement HO ou CD par rapport à la réplication. Le profil moyen du signal de DRIP-Seq dans ces zones confirme que la rencontre frontale des deux polymérases favorise la formation de R-loops. On n'observe cependant pas de diminution du signal pour les rencontres CD. Le profil du DRIP-Seq dans l'orientation CD monte également, mais avec une pente beaucoup plus douce que pour les régions HO. Dans ce contexte, le signal de pRPA-S33 montre cette fois un profil très différent. On observe ainsi une augmentation du signal dans les zones étudiées, ce signal étant plus fort dans les régions HO que CD.

3.3. La position des R-loops sur le gène

La majorité des R-loops se forment sur le corps des gènes, mais l'enrichissement au promoteur et dans la partie terminale est plus important (Ginno et al. 2012, Ginno et al. 2013). Le signal d'arrêt des fourches se concentre sur les gènes fortement exprimés, et plus particulièrement au site de terminaison de ces gènes. On observe également, dans nos cellules déplétées, un fort biais du signal vers les gènes formant déjà des R-loops dans le contrôle, ce qui indique que les hybrides ayant un rôle physiologique peuvent devenir une source de stress si leur stabilité augmente et si une position accumule trop de R-loops.

Plusieurs phénomènes peuvent expliquer pourquoi le signal de stress s'accumule principalement en fin de gène. Dans le cas d'une collision frontale, plus dangereuse pour la stabilité du génome (Prado et Aguilera 2005, Lang et al. 2017), la fourche de réplication rencontre en premier la zone de terminaison du gène, ce qui permet d'expliquer ce biais. Dans les gènes où les R-loops jouent un rôle dans la terminaison de la transcription, l'arrivée de la fourche pourrait créer un stress topologique qui bloque la RNA Pol II et stabilise les hybrides.

De plus, le signal de stress que l'on observe dans nos cellules est concentré sur les gènes fortement exprimés. Sur ces gènes, il y a généralement plusieurs polymérases engagées dans la transcription. Il est possible qu'un conflit en fin de gène, qui bloque l'avancée des polymérases, entraine l'arrêt de plusieurs d'entre elles. Dans cette situation, le stress topologique s'accumule, favorisant d'autant plus la formation de R-loops. Cela pourrait également expliquer pourquoi, dans les cellules déplétées pour ASF/SF2 ou TOP1, une grande partie du signal est distribué autour de positions ou les R-loops sont détectées dans le contrôle.

Ces particularités de la terminaison des gènes pourraient expliquer pourquoi les R-loops en fin de gènes sont plus toxiques que les R-loops au promoteur. Cependant, le signal de pRPA-S33 indique uniquement l'arrêt des fourches. Une situation dangereuse pour la cellule, car le réplisome est instable quand il est bloqué, mais qui peut être résolue et n'entraine pas automatiquement de cassure de l'ADN. Pour observer l'instabilité génomique dans nos cellules, nous avons utilisé le ChIP-Seq de γ-H2AX. Le signal que l'on observe est concentré dans les zones marquées par le DRIP-Seq et le ChIP-Seq de pRPA. Mais la résolution des régions enrichies ne permet pas de déterminer la position exacte à l'origine du stress. De plus, comme les zones touchées sont souvent riches en gènes, il n'est pas possible de déterminer les positions précises à l'origine du stress.

Pour pallier ce manque de précision, nous allons réaliser des expériences de BLESS (direct in situ breaks labeling, enrichment on streptavidin and next-generation sequencing) dans nos cellules (Crosetto et al. 2013). Avec cette technique, les cassures double-brin de l'ADN sont associées à une amorce couplée à une biotine directement dans le noyau de la cellule. L'ADN est ensuite extrait et fragmenté, et les fragments marqués sont capturés grâce à la biotine. Une deuxième amorce est associée à l'extrémité libre des fragments. L'ensemble est ensuite amplifié et séquencé. Les amorces ont une séquence spécifique (barcode) qui permet de déterminer l'orientation des extrémités des cassures. Les données, une fois alignées, permettent de visualiser les sites de cassure au nucléotide près. Associée à nos résultats, cette technique permettra de définir si les R-loops sont bien à l'origine de cassures de l'ADN et si l'augmentation du signal de γ-H2AX dans les cellules shTOP1 est provoquée par des conflits entre la réplication et la transcription.

CONCLUSION

Au début de ce projet, il y quatre ans, nous nous sommes posé une question : comment la dérégulation des oncogènes entraîne le stress réplicatif et l'instabilité génomique dans les cellules précancéreuses?

Nous avons émis l'hypothèse que la transcription pouvait être la source majeure du stress réplicatif. Nous savions que la formation de nouvelles R-loops, liée à la transcription, était à l'origine du ralentissement des fourches de réplication et de l'apparition de l'instabilité génomique, dans les cellules déplétées pour ASF/SF2 et TOP1. Mais les mécanismes qui lient la formation des R-loops au stress réplicatif restaient inconnus.

Les travaux réalisés au cours de ma thèse ont permis de mieux comprendre le rôle des R-loops dans la formation du stress réplicatif. L'étude à l'échelle du génome a confirmé l'hypothèse selon laquelle les hybrides se formaient principalement sur les gènes, et plus particulièrement sur ceux qui ont un haut niveau d'expression et une réplication précoce. Nous avons observé que dans les cellules déplétées pour ASF/SF2 et TOP1, favorisant la formation des R-loops, celles-ci s'accumulent principalement aux positions où elles sont déjà détectées dans les cellules contrôles. De nouvelles positions sont aussi touchées, on les trouve principalement sur d'autres gènes dont le niveau d'expression est un peu plus faible. Leur distribution sur le corps du gène reste identique, indiquant qu'elles se forment principalement aux positions où les R-loops ont un rôle au cours de la transcription. Peut-être que ces transcrits forment des R-loops en conditions normales dans d'autres types cellulaires où leur niveau d'expressions est plus élevé. Dans cet objectif, nous pourrons utiliser le modèle des Lymphocytes B mémoire. Ces cellules sont à l'origine du myélome multiple, elles accumulent des réarrangements chromosomiques qui contribuent au développement de ce cancer. Nos collègues de l'équipe J. Moreaux (CHU Montpellier) ont développé un système de différentiation plasmocytaire *ex vivo* qui permettra l'étude des R-loops dans un contexte physiologique. Au stade préplasmablastes, ces cellules entre en phase de prolifération et de transcription intense, augmentant les risques de stress réplicatif et d'instabilité génomique. Nos résultats préliminaires ont montré que ces cellules accumulent les R-loops spécifiquement au cours de cette étape. L'utilisation du DRIP-Seq nous permettra de déterminer si les R-loops s'accumulent spécifiquement aux positions des translocations caractéristiques du myélome multiple.

L'étude du signal de pRPA-S33 en relation avec celui des R-loops a montré pour la première fois que les fourches de réplication s'arrêtent aux positions ou se forment les

hybrides. Le γ-H2AX, qui s'accumule plus fortement à ces positions dans notre lignée shTOP1, laisse supposer que l'addition de stress topologique est nécessaire pour provoquer des cassures de l'ADN et donc l'instabilité génomique. Les résultats de cette étude montrent que la transcription, à travers les R-loops, provoque l'arrêt des fourches de réplication et peut être une source majeure du stress réplicatif. Le signal de γ-H2AX n'offrant pas une précision suffisante pour conclure que l'ADN se casse directement au niveau des R-loops, nous prévoyons d'utiliser la technique de BLESS qui va nous permettre d'étudier les cassures de l'ADN dans nos cellules avec une précision de quelques nucléotides.

ANNEXES

Annexe 1 : Exemple de fichier de configuration

```
#Authors:  Ismael Padioleau
#May 3, 2014
#Default configuration for the rnaseq pipeline with comments
#This file is a comma separated value file, lines starting with "#" are comments not readed by
the pipeline.
#Fist words are key words that must not be changed.
#Email of the data managers
#This option is a list of email to which report will be sent during the data processing
#The reports are sent at different steps of the pipeline and keep user informed of the
processus

contact_email,ismael.padioleau@igh.cnrs.fr

#Are reads Pair-end read: 1 if they are pair-end, 0 if they are single-end
#This is a boolean use to determine if the run is paired-end or not
pairend,0

#reference genome used to map reads
#The reference genome is used by the mapper to aligned reads on genome.
# --WARNING--  Make sure that the reference correspond to the mapper you decided to use.
ref_genome_name,/save/ipadioleau/indexes/hg19_selection/hg19.fa

#Choose your mapper (bwa, bowtie)
mapper,bowtie

#required command and options for the mapper (here bowtie). '-p' and '-S' option are already
set you must not set it again.
#bowtie,

#Use rmdup with samtools if uncommented. Options for samtools rmdup can be given after comma.
#samtools_rmdup_options,-s

#Limit in distance between reads for the summary file
#dist_summary_upper_limit,150

# Bin size for bam to bigwig with deeptools
binSize_bigwig,1
fragmentLength,51
deeptools_format,bigwig

# peak calling soft macs or sicer
peak_caller,macs

## Only one can be uncommented SICER or MACS options
# Options for macs. -g must be provided with hs for human and 9996000 for yeast.
macs_option, -g hs --keep-dup 5 -q 0.05

# Options SICER, must be given in the format below (Default value) where genome = hg19,
redundancy = 1, windows = 500, fragment size = 200, effective genome fraction = 0.74, gap size
(bp) = 1000, FDR = .01
#sicer_option, hg38 5 1500 200 0.74 4500 0.05

## Bam coverage options
# annotation dir is required
# mulitBamCov_option is optional
annotation_dir,/home/ipadioleau/work/170410_new_pipeline/bed
#mulitBamCov_option,-s

# Bam correlation and PCA (deeptools)
# Be carefull, if you change option for multiBamSummary or plotCorrelation, you have to give
mandatory options
# Default is :
# for multiBamSummary_option : ' bins -binSize 50 '
# for plotCorrelation_option : ' -corMethod spearman -whatToPlot heatmap '
# If you change these option uncomment the following lines and be sure to give values to these
mandatory arguments.
#multiBamSummary_option,bins -binSize 5000
plotCorrelation_option,--corMethod spearman --whatToPlot scatterplot
#plotPCA_option,

##############################################################################################
#                                  SOFTWARE PATH                                             #
##############################################################################################
#This part, in coma separated value format, is used by main.py to set executable directory
#In each lane, the first word is a keyword that should not be changed, the second part after
#the comma is the path to the corresponding executable.
#path to python 3.2
```

```
python,/usr/local/bioinfo/src/python/current/bin/python

#Path to bowtie
bowtie_dir,/usr/local/bioinfo/bin/bowtie2

#Path to samtools
samtools,/usr/local/bioinfo/bin/samtools

#bamToFastq
#bamToFastq,/data/2backup/common/tools/src/Hydra-Version-0.5.3/bin/bamToFastq

#macs
macs_dir,/usr/local/bioinfo/src/python/current/bin/macs2

#sicer
sicer_dir,/home/ipadioleau/work/opt/SICER1.1/SICER/SICER.sh

#bedtools
bedtools_dir,/usr/local/bioinfo/bin/bedtools
```

Annexe 2 : Code du pipeline

Le code principal est contenu dans le script « main.py ». Il peut appeler un ensemble de fonctions contenues dans d'autres scripts. Le nom de chaque fichier est indiqué en noir, avant le code qu'il contient.

La première partie du code du fichier main.py sert à indiquer où se trouvent les autres fichiers. En effet, pour simplifier la lecture du code, il est séparé en différents fichiers qui sont eux-mêmes regroupés dans différents dossiers. Il y a par exemple un dossier pour toute la partie alignement des données, un autre pour le contrôle qualité. Cette organisation permet à la personne qui programme de retrouver facilement les fonctions qui concernent un logiciel.

```
1.  #!/usr/local/bioinfo/src/python/current/bin/python
2.  # Author: Ismael Padioleau
3.  # Contact: ismael.padioleau@igh.cnrs.fr
4.  # May 15th 2014
5.  # main script of the pipeline to process illumina sequencer data
6.
7.  ##################
8.  #                #
9.  #    Imports     #
10. #                #
11. ##################
12. import os
13. import sys
14. import uuid
15. import argparse
16. import fnmatch
17.
18. # Import our own functions
19. pipeline_path = sys.path[0]
20. pipeline_tools_path = os.path.abspath(pipeline_path + '/pipeline_tools/')
21. mapping_path = os.path.abspath(pipeline_path + '/mapping/')
22. quantification_path = os.path.abspath(pipeline_path + '/quantification/')
23. qc_summary_and_graph_path = os.path.abspath(pipeline_path + '/qc_summary_and_graph/')
24. backup_path = os.path.abspath(pipeline_path + '/backup/')
25. peak_calling_path = os.path.abspath(pipeline_path + '/peak_calling/')
26. coverage = os.path.abspath(pipeline_path + '/coverage/')
27. deeptools = os.path.abspath(pipeline_path + '/deeptools/')
28. sys.path.append(pipeline_tools_path)
29. sys.path.append(mapping_path)
30. sys.path.append(peak_calling_path)
31. sys.path.append(quantification_path)
32. sys.path.append(qc_summary_and_graph_path)
33. sys.path.append(coverage)
34. sys.path.append(deeptools)
35. from configParser import getConfigDict
36. from runFastqToMap_bowtie import fastqToBam_bowtie
37. from runFastqToMap_samtools import samToBam
38. from runFastqToMap_samtools import indexBam
39. from runFastqToMap_samtools import sortBam
40. from runFastqToMap_deepTools import bamToBigWig
41. from runFastqToMap_deepTools import bamRatio
42. from runFastqToMap_bedtools import bamToBed
43. from runPeakCalling_macs import peakCalling_macs
44. from runPeakCalling_sicer import peakCalling_sicer
45. from coverage_multiBamCov import multiBamCov
46. from deeptools_multiBamSummary import multiBamSummary
47. from runMappingQuality import runMappingQuality
48. from runFastqQuality import runFastqQuality
49. from writeLogs import writeLog, writeEmail
50.
51. ####################
52. #                  #
53. #    Functions     #
54. #                  #
55. ####################
56.
57. # Function to check directories given as input
58. # If multiple directories are given as input, do a temporary one with all files in it as symbo
    lic links
59. def check_input_dir(type, dict):
60.     info_dict = {'fastq':['-
    fastq','args.fastq_dir','.fastq.gz','fastq_list','fastq_dir'], 'sam':['-
```

```
    sam','args.sam_dir','.sam','sam_list','sam_dir'], 'bam':['-
    bam','args.bam_dir','.bam','bam_list','bam_dir'], 'bed':['-
    bed','args.bed_dir','.bed','bed_list','bed_dir'], 'qc_fastq':['-
    qcf','args.qcf_dir','_qc_stats','qc_fastq_list','qc_fastq_dir'], 'qc_mapping':['-
    qcm','args.qcm_dir','.csv','qc_mapping_list','qc_mapping_dir']}
61.     type_option = info_dict[type][0]
62.     type_arg = info_dict[type][1]
63.     type_file = info_dict[type][2]
64.     type_list = info_dict[type][3]
65.     type_dir = info_dict[type][4]
66.     #check if the directory exist.
67.     if not os.path.isdir(dict[type_dir]):
68.         sys.stderr.write('ERROR: You must provide an correct ' + type + ' directory to run you
    r task : arg ' + type_option  + '\n')
69.         sys.exit(1)
70.
71.     #if multiple input directories, create a temporary directory with symbolic links to all da
    ta.
72.     if eval(type_arg) and len(eval(type_arg)) > 1:
73.         tmp_dir = eval(type_arg)[0] + '/tmp'
74.         dict[type_dir] = tmp_dir
75.         dict['tmp_dirs'].append(tmp_dir)
76.         if not os.path.isdir(tmp_dir):
77.             os.makedirs(tmp_dir)
78.             for dir  in  eval(type_arg):
79.                 if not os.path.isdir(dir):
80.                     sys.stderr.write('ERROR: You must provide an correct ' + type + ' director
    y to run your task : arg ' + type_option  + '\n')
81.                     sys.exit(1)
82.                 os.system('ln -
    s ' + os.path.abspath(dir) + '/*' + type_file + ' ' + tmp_dir)
83.
84.     #check sample list
85.     if not type_list in dict:
86.         dict[type_list] = []
87.         if type == 'fastq' or type == 'qc_fastq':
88.             type_file = '_1' + type_file
89.         for file in os.listdir(dict[type_dir]):
90.             if fnmatch.fnmatch(file, '*' + type_file):
91.                 basename = file.split(type_file)[0]
92.                 if basename in dict[type_list]:
93.                     sys.stderr.write('You provided multiple sample directories and some contai
    n identical file (name), to run the pipeline make sure that each sample as a unique name.\n')

94.                     sys.exit(1)
95.                 dict[type_list].append(basename)
96.     return(dict)
97.
98. # Function to remove temporary directory created by the pipeline
99. def clean_tmp_dir(configDict):
100.     for dir in configDict['tmp_dirs']:
101.         ### WARNING THIS IS SET TO DELETE DIRECTORIES, ALWAYS MAKE SURE THAT NOTHING IMPORTANT
     SI SENT TO THIS FUNCTION
102.         info = dir.split('/')
103.         if 'tmp' in info: #A minimum check to be sure that we don't try to delete '/' or '/dat
    a' or anything that does not contain 'tmp' in it's path
104.             command = 'bsub -o /dev/null -w \'ended(' + configDict['uid'] + '*)\' rm -
    r ' + dir
105.             print('Clean tmp directory for project : ' + configDict['project'])
106.             os.system(command)
107.
108.#######################
109.#                     #
110.#    Get arguments    #
111.#                     #
112.#######################
```

```
113.
114.parser = argparse.ArgumentParser(description='Pipeline to process sequencing data.')
115.parser.add_argument('-
    v', dest='version', action='store_true', help='Display pipeline version')
116.#If user is asking for version
117.if len(sys.argv) > 1:
118.    if sys.argv[1] == '-v':
119.        print('Pipeline version 1.00\n')
120.        sys.exit(0)
121.
122.parser.add_argument('-fastq', '-
    fastqdir', dest='fastq_dir', type=str, nargs='+', help='Absolut path fastq to diretor(y)ies. I
    f multiple directories, separate eache path with space')
123.parser.add_argument('-sam', '-
    samdir', dest='sam_dir', type=str, nargs='+', help='Path sam diretory, is multiple, separate w
    ith space.')
124.parser.add_argument('-bam', '-
    bamdir', dest='bam_dir', type=str, nargs='+', help='Path bam diretory, is multiple, separate w
    ith space.')
125.parser.add_argument('-bed', '-
    beddir', dest='bed_dir', type=str, nargs='+', help='Path bed diretory, is multiple, separate w
    ith space.')
126.parser.add_argument('-ratio', '-
    ratiodir', dest='ratio_dir', type=str, nargs='+', help='Path ratio diretory, is multiple, sepa
    rate with space.')
127.parser.add_argument('-
    sortedBam', dest='sorted_bam_dir', type=str, nargs=1, help='Path to sorted bam directory, if n
    ot set, first bam directory is used.')
128.parser.add_argument('-
    splitMapping', dest='split_mapping_dir', type=str, nargs=1, help='Path to split mapping direct
    ory.')
129.parser.add_argument('-
    indexBam', dest='index_bam_dir', type=str, nargs=1, help='Path to index bam directory, if not
    set, first bam directory is used.')
130.parser.add_argument('-qcf', '-
    qc_fastq_dir', dest='qcf_dir', type=str, nargs='+', help='Path qc_fastq diretory')
131.parser.add_argument('-qcm', '-
    qc_mapping_dir', dest='qcm_dir', type=str, nargs='+', help='Path qc_mapping  diretory')
132.parser.add_argument('-od', '-
    outputdir', dest='output_dir', type=str, nargs=1, help='Path to output directory')
133.parser.add_argument('-d', dest='date', type=str, nargs=1, help='Date of the run: YYMMDD')
134.parser.add_argument('-
    cf', dest='config_file_path', type=str, nargs=1, required=True, help='Path to your configurati
    on file')
135.parser.add_argument('-
    pn', dest='project_name', type=str, nargs='+', help='List of project to process, must be in sa
    mplesheet')
136.parser.add_argument('-
    pair', dest='pair_file_path', type=str, nargs=1, help='Path to a tab separated file with pair
    of IP/Input to be use in peak calling step')
137.parser.add_argument('-t', dest='task', type=str, required=True, nargs='+', help='')
138.
139.####################
140.#                  #
141.#    CHECK ARGS    #
142.#                  #
143.####################
144.
145.# Get command line args
146.args = parser.parse_args()
147.
148.# Get list of tasks
149.if not args.task:
150.    sys.stderr.write('ERROR: You must give task(s) to run : arg -t\n')
151.    sys.exit(1)
152.task_list = args.task
```

```
if 'all' in task_list:
    task_list = ['1','2','3','4','5','6','7','8']

# Check and parse config file
dict_of_configDict = {}
if not args.config_file_path:
    if not args.config_file_list_path:
        sys.stderr.write('ERROR: You must provide a configuration file to run the pipeline : arg -cf or -cfl\n')
        sys.exit(1)
    else:
        FILE = open(os.path.abspath(args.config_file_list_path[0]),'r')
        for line in FILE:
            info = line.strip().split(',')
            config_file_path = os.path.abspath(info[1])
            if not (os.path.isfile(config_file_path)):
                sys.stderr.write('ERROR: The configuration file you gave doesn\'t exist: [ ' + config_file_path + ' ]\n')
                sys.exit(1)
            dict_of_configDict[info[0]] = getConfigDict(config_file_path)
            dict_of_configDict[info[0]]['config_file_path'] = config_file_path
    configDict = {}
    for key in dict_of_configDict[info[0]]:
        configDict[key] = dict_of_configDict[info[0]][key]
else:
    config_file_path = os.path.abspath(args.config_file_path[0])
    if not (os.path.isfile(config_file_path)):
        sys.stderr.write('ERROR: The configuration file you gave doesn\'t exist: [ ' + config_file_path + ' ]\n')
        sys.exit(1)
    configDict = getConfigDict(config_file_path)
    configDict['config_file_path'] = config_file_path

# Create a unique ID for this run
configDict['uid'] = (str(uuid.uuid1())[:8])

# Hardcoded script directories
configDict['qc_fastq_script'] = pipeline_path + '/qc_summary_and_graph/qc_fastq_script/qc_fastq'
configDict['qc_mapping_script'] = pipeline_path + '/qc_summary_and_graph/get_mapping_info.py'
configDict['qc_summary_script'] = pipeline_path + '/qc_summary_and_graph/do_summary_table.py'
configDict['distance_between_reads_script'] = pipeline_path + '/qc_summary_and_graph/distance_between_reads.py'
configDict['distance_between_reads_summary'] = pipeline_path + '/qc_summary_and_graph/distance_between_reads_summary.py'
configDict['graph_fastq_script'] = pipeline_path + '/qc_summary_and_graph/pipelineFastq_qc_Plots.R'
configDict['graph_mapping_script'] = pipeline_path + '/qc_summary_and_graph/pipelineMapping_qc_Plots.R'
# Add complementary values in config dict
configDict['tmp_dirs'] = []
configDict['contact_email'] = configDict['contact_email'].rsplit(" ")
configDict['pipeline_path'] = pipeline_path
configDict['task'] = task_list

#####################
#                   #
#    Start tasks    #
#                   #
#####################

print('Start\n')
print('Unique ID of this run: ' + str(configDict['uid']) + '\n')

```

```
####                      ####
####    Set directories   ####
####                      ####
####
# Project name is given by user, or set to No_name_provided
if not 'projects' in configDict:
    if not args.project_name:
        configDict['projects'] = ['No_name_provided']
    else:
        configDict['projects'] = args.project_name

# Prepare directories info
for project in configDict['projects']:
    configDict['project'] = project
    if args.fastq_dir:
        configDict['fastq_dir'] = os.path.abspath(args.fastq_dir[0])
    elif args.output_dir:
        configDict['fastq_dir'] = os.path.abspath(args.output_dir[0] + '/fastq/')
    if args.sam_dir:
        configDict['sam_dir'] = os.path.abspath(args.sam_dir[0])
    elif args.output_dir:
        configDict['sam_dir'] = os.path.abspath(args.output_dir[0] + '/sam/')
    if args.bam_dir:
        configDict['bam_dir'] = os.path.abspath(args.bam_dir[0])
    elif args.output_dir:
        configDict['bam_dir'] = os.path.abspath(args.output_dir[0] + '/bam/')
    if args.output_dir:
        configDict['profileDir'] = os.path.abspath(args.output_dir[0] + '/profile_dir/')
    if args.output_dir:
        if 'peak_caller' in configDict and configDict['peak_caller'] == 'macs':
            configDict['peak_dir'] = os.path.abspath(args.output_dir[0] + '/macs/')
        elif 'peak_caller' in configDict and configDict['peak_caller'] == 'sicer':
            configDict['peak_dir'] = os.path.abspath(args.output_dir[0] + '/sicer/')
    if args.ratio_dir:
        configDict['ratio_dir'] = os.path.abspath(args.ratio_dir[0])
    elif args.output_dir:
        configDict['ratio_dir'] = os.path.abspath(args.output_dir[0] + '/ratio/')
    if args.sorted_bam_dir:
        configDict['sort_bam_dir'] = os.path.abspath(args.sorted_bam_dir[0])
    elif 'bam_dir' in configDict:
        configDict['sort_bam_dir'] = configDict['bam_dir']
    elif args.output_dir:
        configDict['sort_bam_dir'] = os.path.abspath(args.output_dir[0] + '/sorted_bam/')
    if args.index_bam_dir:
        configDict['index_bam_dir'] = os.path.abspath(args.index_bam_dir[0])
    elif 'bam_dir' in configDict:
        configDict['index_bam_dir'] = configDict['bam_dir']
    elif args.output_dir:
        configDict['index_bam_dir'] = os.path.abspath(args.output_dir[0] + '/index_bam/')
    if args.bed_dir:
        configDict['bed_dir'] = os.path.abspath(args.bed_dir[0])
    elif args.output_dir:
        configDict['bed_dir'] = os.path.abspath(args.output_dir[0] + '/bed/')
    if args.qcf_dir:
        configDict['qc_fastq_dir'] = os.path.abspath(args.qcf_dir[0])
    elif args.output_dir:
        configDict['qc_fastq_dir'] = os.path.abspath(args.output_dir[0] + '/qc_fastq/')
    if args.qcm_dir:
        configDict['qc_mapping_dir'] = os.path.abspath(args.qcm_dir[0])
    elif args.output_dir:
        configDict['qc_mapping_dir'] = os.path.abspath(args.output_dir[0] + '/qc_mapping/')
    if '8' in task_list:
        configDict['count_dir'] = os.path.abspath(args.output_dir[0] + '/count/')
    if '9' in task_list:
        configDict['mutliBamSum_dir'] = os.path.abspath(args.output_dir[0] + '/mutliBamSum/')
```

```
    if args.output_dir:
        configDict['info_dir'] = os.path.abspath(args.output_dir[0] + '/info/')

    ####
    ######################################################
    #                                                    #
    #    Tasks 1, from fastq to sorted bam, and more     #
    #                                                    #
    ######################################################

    if('1' in task_list or '1.1' in task_list):
        if 'mapper' not in configDict:
            sys.stderr.write('ERROR: You must provide a mapper name in the configuration file to run mapping \n')
            sys.exit(1)
        else:
            mapper = configDict['mapper']
        #################
        #               #
        #    BOWTIE2    #
        #               #
        #################
        if mapper == 'bowtie':
            if('1' in task_list or '1.1' in task_list):
                # Set a mark in configdict to show that 1.1 run
                configDict['t1.1_run'] = ''
                ###                  ###
                ###   FASTQ TO BAM   ###
                ###                  ###
                # Check directories
                if not 'fastq_dir' in configDict:
                    sys.stderr.write('ERROR: You must provide a fastq directory to run bowtie2 task : arg -fastq\n')
                    sys.exit(1)
                if not 'bam_dir' in configDict:
                    sys.stderr.write('ERROR: You must provide a bam directory to run bowtie2 task : arg -bam\n')
                    sys.exit(1)
                if os.path.isdir(configDict['bam_dir']):
                    sys.stderr.write('ERROR: The bam directory you provide must be new, we cowardly refuse to take risk of overwriting existing data\n')
                    sys.exit(1)
                if os.path.isdir(configDict['profileDir']):
                    sys.stderr.write('ERROR: The profileDir directory you provide must be new, we cowardly refuse to take risk of overwriting existing data\n')
                    sys.exit(1)
                os.makedirs(configDict['profileDir'])
                configDict = check_input_dir('fastq',configDict)
                #Run bowtie
                configDict = fastqToBam_bowtie(configDict)

    ####
    ################
    #              #
    #   SAMtools   #
    #              #
    ################
    ###                               ###
    ###   SAM TO BAM SORTED + INDEX   ###
    ###                               ###
    if('1.2' in task_list):
        configDict['t1.2_run'] = ''
        # Check directories
        if 'sam_dir' not in configDict:
            sys.stderr.write('ERROR: You must provide a sam directory to run sam to bam task : arg -sam\n')
```

```
            sys.exit(1)
        if 'bam_dir' not in configDict:
            sys.stderr.write('ERROR: You must provide a bam directory to run sam to bam task : arg -bam\n')
            sys.exit(1)
        if  os.path.isdir(configDict['bam_dir']):
            sys.stderr.write('ERROR: The bam directory you provide must be new, we cowardly refuse to take risk of overwriting existing data\n')
            sys.exit(1)
        os.makedirs(configDict['bam_dir'])
        # Check input
        configDict = check_input_dir('sam',configDict)
        # Run sam to bam
        configDict = samToBam(configDict)
    ###               ###
    ###   SORT BAM    ###
    ###   INDEX BAM   ###
    ###               ###
    if('1.3' in task_list or '1.4' in task_list ):
        # Check directories
        if 'bam_dir' not in configDict:
            sys.stderr.write('ERROR: You must provide a bam directory to run sam to bam task : arg -bam\n')
            sys.exit(1)
        configDict = check_input_dir('bam',configDict)
        # If not done create sorted bam directory and index for bam directory
        if not os.path.isdir(configDict['sort_bam_dir']):
            os.makedirs(configDict['sort_bam_dir'])
        if not os.path.isdir(configDict['index_bam_dir']):
            os.makedirs(configDict['index_bam_dir'])
        if 't1.2_run'  in configDict:
            configDict['bam_list'] = configDict['sam_list']
        # Run bam sorting
        if('1.3' in task_list):
            configDict['t1.3_run'] = ''
            configDict = sortBam(configDict)
        #run bam indexing
        if('1.4' in task_list):
            configDict['t1.4_run'] = ''
            if args.index_bam_dir:
                configDict = indexBam(configDict)
            else:
                if args.bam_dir:
                    backup_bam_dir = configDict['bam_dir']
                    configDict['multiple_index_bam_dir'] = ''
                    for dir in args.bam_dir:
                        configDict['index_bam_dir'] = dir
                        configDict['bam_dir'] = dir
                        configDict = indexBam(configDict)
                    configDict['bam_dir'] = backup_bam_dir
                else:
                    configDict = indexBam(configDict)

    ###              ###
    ###   QC FASTQ   ###
    ###              ###
    if('2' in task_list):
        #Tag to say that this task run
        configDict['t2_run']=''
        if 'fastq_dir' not in configDict:
            sys.stderr.write('ERROR: You must provide a fastq directory to run fastq QC : arg -fastq\n')
            sys.exit(1)
        if 'qc_fastq_dir' not in configDict:
            sys.stderr.write('ERROR: You must provide a QC fastq directory to run fastq QC : arg -qcf\n')
```

```
395.            sys.exit(1)
396.        #Create symbolic links if more than one input directory
397.        configDict = check_input_dir('fastq',configDict)
398.        #run qc on fastq
399.        configDict = runFastqQuality(configDict)
400.##
401.#
402.#
403.#     CONTINUE EDITTING
404.#
405.#
406.    ###                ###
407.    ###   QC MAPPING   ###
408.    ###                ###
409.    if('3' in task_list ):
410.        # Tag to say that this task run
411.        configDict['t3_run']=''
412.        if 'bam_dir' not in configDict:
413.            sys.stderr.write('ERROR: You must provide a bam directory to run mapping QC : arg
    -bam\n')
414.            sys.exit(1)
415.        if 'qc_mapping_dir' not in configDict:
416.            sys.stderr.write('ERROR: You must provide a QC mapping directory to run mapping QC
     : arg -qcm\n')
417.            sys.exit(1)
418.        if  os.path.isdir(configDict['qc_mapping_dir']):
419.            sys.stderr.write('ERROR: The qc_mapping directory you provide must be new, we cowa
    rdly refuse to take risk of overwriting existing data\n')
420.            sys.exit(1)
421.        os.makedirs(configDict['qc_mapping_dir'])
422.        # Create symbolic links if more than one input directory
423.        if 't1.1_run' in configDict:
424.            configDict['bam_list'] = configDict['fastq_list']
425.        elif 't1.2_run' in configDict:
426.            configDict['bam_list'] = configDict['sam_list']
427.        else:
428.            configDict = check_input_dir('bam',configDict)
429.        # Run qc on mapping
430.        configDict = runMappingQuality(configDict)
431.        #configDict = runSummaryTable(configDict)
432.
433.    ####
434.    ##################
435.    #                #
436.    #   deepTools    #
437.    #                #
438.    ##################
439.    if('4' in task_list):
440.        configDict['t4_run'] = ''
441.        # Check directories
442.        if 'bam_dir' not in configDict:
443.            sys.stderr.write('ERROR: You must provide a bam directory to run bam to bigwig tas
    k : arg -bam\n')
444.            sys.exit(1)
445.#        if  os.path.isdir(configDict['profileDir']):
446.#            sys.stderr.write('ERROR: The bigWig directory you provide must be new, we cowardl
    y refuse to take risk of overwriting existing data\n')
447.#            sys.exit(1)
448.        if not os.path.isdir(configDict['profileDir']):
449.            os.makedirs(configDict['profileDir'])
450.        # Check input
451.        configDict = check_input_dir('bam',configDict)
452.        # Run bamToBigwig
453.        configDict = bamToBigWig(configDict)
454.
455.    ####
```

```
    ################
    #              #
    #   bedTools   #
    #              #
    ################
    if('5' in task_list):
        configDict['t5_run'] = ''
        # Check directories
        if 'bam_dir' not in configDict:
            sys.stderr.write('ERROR: You must provide a bam directory to run bam to bigwig tas
    k : arg -bam\n')
            sys.exit(1)
        if  os.path.isdir(configDict['bed_dir']):
            sys.stderr.write('ERROR: The bed directory you provide must be new, we cowardly re
    fuse to take risk of overwriting existing data\n')
            sys.exit(1)
        os.makedirs(configDict['bed_dir'])
        # Check input
        configDict = check_input_dir('bam',configDict)
        # Run bamToBigwig
        configDict = bamToBed(configDict)

    ####
    ####################
    #                  #
    #   Peak calling   #
    #                  #
    ####################
    if(('6' in task_list) and ('peak_caller' in configDict)):
        configDict['t6_run'] = ''
        if not args.pair_file_path:
            sys.stderr.write('ERROR: You must provide a pair file to run peak calling task : a
    rg -
    pair\nThis is a tab separated file that contain pair of sample to use in peak calling\n')
            sys.exit(1)
        configDict['pair_file'] = os.path.abspath(args.pair_file_path[0])
        # Check directories
        pair_file_reader = open(configDict['pair_file'], 'r')
        pair_info = []
        for line in pair_file_reader:
            tmp_info_pair = line.strip().split('\t')
            pair_info.append(tmp_info_pair)
        pair_file_reader.close()
        configDict['pair_list'] = pair_info
        if  os.path.isdir(configDict['peak_dir']):
            sys.stderr.write('ERROR: The peak directory you provide must be new, we cowardly r
    efuse to take risk of overwriting existing data\n')
            sys.exit(1)
        os.makedirs(configDict['peak_dir'])
        if 'macs_option' in configDict:
            if 'bam_dir' not in configDict:
                sys.stderr.write('ERROR: You must provide a bam directory to run peak calling
    with macs task : arg -bam\n')
                sys.exit(1)
            configDict = check_input_dir('bam',configDict)
            configDict = peakCalling_macs(configDict)
        elif 'sicer_option' in configDict:
            if 'bed_dir' not in configDict:
                sys.stderr.write('ERROR: You must provide a bed directory to run peak calling
    task with sicer: arg -bed\n')
                sys.exit(1)
            configDict = check_input_dir('bed',configDict)
            configDict = peakCalling_sicer(configDict)
        else:
            sys.stderr.write('ERROR: You must provide peak calling option in you configuaratio
    n file. For macs or sicer.\n')
```

```
514.            sys.exit(1)
515.    ####
516.    ####################
517.    #                  #
518.    #   Ratio          #
519.    #                  #
520.    ####################
521.    if(('7' in task_list) and args.pair_file_path):
522.        configDict['t7_run'] = ''
523.        configDict['pair_file'] = os.path.abspath(args.pair_file_path[0])
524.        # Check directories
525.        pair_file_reader = open(configDict['pair_file'], 'r')
526.        pair_info = []
527.        for line in pair_file_reader:
528.            tmp_info_pair = line.strip().split('\t')
529.            pair_info.append(tmp_info_pair)
530.        pair_file_reader.close()
531.        configDict['pair_list'] = pair_info
532.        if  os.path.isdir(configDict['ratio_dir']):
533.            sys.stderr.write('ERROR: The ratio directory you provide must be new, we cowardly
    refuse to take risk of overwriting existing data\n')
534.            sys.exit(1)
535.        os.makedirs(configDict['ratio_dir'])
536.        if 'bam_dir' not in configDict:
537.            sys.stderr.write('ERROR: You must provide a bam directory to run ratio task : arg
    -bam\n')
538.            sys.exit(1)
539.        configDict = check_input_dir('bam',configDict)
540.        configDict = bamRatio(configDict)
541.    ####
542.    ####################
543.    #                  #
544.    #   multiBamCov    #
545.    #                  #
546.    ####################
547.    if('8' in task_list):
548.        configDict['t8_run'] = ''
549.        # Check directories
550.        if 'bam_dir' not in configDict:
551.            sys.stderr.write('ERROR: You must provide a bam directory to run bam to bigwig tas
    k : arg -bam\n')
552.            sys.exit(1)
553.        if 'annotation_dir' not in configDict:
554.            sys.stderr.write('ERROR: You must provide an annotation (with bed files) directory
     to run multiBamCov task : annotation_dir in configuration file\n')
555.            sys.exit(1)
556.        if not os.path.isdir(configDict['count_dir']):
557.            os.makedirs(configDict['count_dir'])
558.        # Check input
559.        configDict = check_input_dir('bam',configDict)
560.        cpt=0
561.        for file in os.listdir(configDict['annotation_dir']):
562.            if fnmatch.fnmatch(file, '*bed'):
563.                cpt+=1
564.        if cpt==0:
565.            sys.stderr.write('ERROR: You must provide a correct annotation directory with at l
    east one BED file (extention must be .bed) to run multiBamCov task : annotation_dir in configu
    ration file\n')
566.            sys.exit(1)
567.        # Run multiBamCov
568.        configDict = multiBamCov(configDict)
569.    ####
570.    ######################
571.    #                    #
572.    #   multiBamSummary  #
573.    #                    #
```

```
    ######################
    if('9' in task_list):
        configDict['t9_run'] = ''
        # Check directories
        if 'bam_dir' not in configDict:
            sys.stderr.write('ERROR: You must provide a bam directory to run bam to bigwig tas
    k : arg -bam\n')
            sys.exit(1)
        if not os.path.isdir(configDict['mutliBamSum_dir']):
            os.makedirs(configDict['mutliBamSum_dir'])
        # Check input
        configDict = check_input_dir('bam',configDict)
        # Run multiBamSummary
        configDict = multiBamSummary(configDict)

    print("\nCreate script for error control")
    script = open(os.path.abspath(args.output_dir[0]) + '/control_script.sh', 'w')
    script.write('#!/bin/bash\n#$ -m a\n#$ -q workq\n')
    script.write('#$ -o control_for_error.out\n')
    script.write('#$ -e control_for_error.err\n')
    script.write('#$ -N control_for_error\n')
    #script.write('#$ -hold_jid job_' + str(configDict['uid']) + '*\n')
    script.write('grep -i -m1 err */log/* > control_for_error.INFO\n')
    script.write('grep -i -m1 zip */log/* > control_for_error.INFO\n')
    script.write('grep -i -m1 fail */log/* > control_for_error.INFO\n')
    script.write('grep -i -m1 kill */log/* > control_for_error.INFO\n')
    script.write('\n#end of the script\n')
    script.close()
    #os.system('qsub ' + os.path.abspath(args.output_dir[0]) + '/control_script.sh')
    ####

print('\nUnique ID of this run: ' + str(configDict['uid']) + '\n')
```

coverage_multiBamCov.py

```
#!/usr/bin/env python3
# Author: Ismael padioleau
# Contact: ismael.padioleau@igh.cnrs.fr
# April 10th 2017
# Calculation on mapped data
# Function exported multiBamCov

import os

####
## FUNCTIONS
#
def multiBamCov_Commands(name, configDict):
    #set variables
    uid = configDict['uid']
    project = configDict['project']
    countDir = configDict['count_dir']
    bamDir = configDict['bam_dir']
    annotationDir = configDict['annotation_dir']
    job_basename = 'job_' + uid + '_multiBamCov_' + name
    scriptDir = countDir + '/script/'
    logDir = countDir + '/log/'
    wait_condition = '#\n'
    #Set waitcondition
    if 't1.1_run' in configDict or 't1.2_run' in configDict:
        wait_condition = '#$ -hold_jid job_' + uid + '_fastqToBam_samToBam_* \n'
    #build directory
    if not os.path.exists(logDir):
       os.makedirs(logDir)
```

```
    if not os.path.exists(scriptDir):
       os.makedirs(scriptDir)
    #get options
    multiBamCov_option = ''
    if 'multiBamCov_option' in configDict:
        multiBamCov_option = configDict['multiBamCov_option']
    #create commands
    command_bam = 'for bed in *bed; do multiBamCov ' + multiBamCov_option + ' -bams ' + name + '.bam -bed $bed > ${bed%%.*}_COV_BY_' + name + '_count.tsv; done\n'
    command_nbRead = 'samtools flagstat ' + name + '.bam | grep "mapped (" | awk -v OFS=\' \' \'{print $1}\' > nbRead\n'
    command_RPKM = 'for tsv in *.tsv; do awk -v OFS="\t" -v mapped="$(<nbRead)" \'{$NF=($NF/(mapped/1000000))/(($3-$2)/1000); print}\' $tsv > ${tsv%%.*}_RPKM.tsv; done\n'
    # Create a unique var name to save temporary directory
    dir_name = name.replace('-','_')
    dir_name = dir_name.replace('.','_')
    dir_name = dir_name.replace(' ','_')
    tmpdir = 'TMPDIR_' + dir_name
    # Write and execute the script
    script = open(scriptDir + name + '_multiBamCov.sh', 'w')
    script.write('#!/bin/bash\n#$ -m a\n#$ -q workq\n')
    script.write('#$ -l h_vmem=10G\n#$ -l mem=10G\n')
    script.write('#$ -o ' + logDir + name + '_multiBamCov.out\n')
    script.write('#$ -e ' + logDir + name + '_multiBamCov.err\n')
    script.write('#$ -N ' + job_basename + '\n')
    script.write(wait_condition)
    script.write(tmpdir + '=$(mktemp -d /tmp/multiBamCov_XXXXXX) \n')
    script.write('cd $' + tmpdir + ' \n')
    script.write('ln -s ' + bamDir + '/' + name + '.bam* .\n')
    script.write('ln -s ' + annotationDir + '/*bed .\n')
    script.write(command_nbRead)
    script.write(command_bam)
    script.write(command_RPKM)
    script.write('mv $' + tmpdir + '/*.tsv ' + countDir + '\n')
    script.write('rm -r $' + tmpdir)
    script.write('\n#end of the script\n')
    script.close()
    os.system('qsub ' + scriptDir + name + '_multiBamCov.sh')


##############
####
## CALL
#
def multiBamCov(configDict):
    print('multiBamCov project : ' + configDict['project'])
    #Call function to run command for each sample
    for name in  configDict['bam_list']:
        multiBamCov_Commands(name, configDict)
    return(configDict)
)
```

deeptools_multiBamSummary.py

```
#!/usr/bin/env python3
# Author: Ismael padioleau
# Contact: ismael.padioleau@igh.cnrs.fr
# April 10th 2017
# Run multi bam summary from deeptools

import os

####
```

```
## FUNCTIONS
#
def multiBamSummary_Commands(configDict):
    #set variables
    bam_list = configDict['bam_list']
    uid = configDict['uid']
    project = configDict['project']
    bamDir = configDict['bam_dir']
    mutliBamSum_dir = configDict['mutliBamSum_dir']
    scriptDir = mutliBamSum_dir + '/script/'
    logDir = mutliBamSum_dir + '/log/'
    job_basename = 'job_' + uid + '_deeptool_multiBamSummary'
    wait_condition = '#\n'
    if 't1.1_run' in configDict or 't1.2_run' in configDict:
        wait_condition = '#$ -hold_jid job_' + uid + '_fastqToBam_samToBam_* \n'
    #build sam directory
    if not os.path.exists(logDir):
       os.makedirs(logDir)
    if not os.path.exists(scriptDir):
       os.makedirs(scriptDir)
    #create commands
    multiBamSummary_option = ' bins --binSize 50 -p 8 '
    if 'multiBamSummary_option' in configDict:
        multiBamSummary_option = configDict['multiBamSummary_option']

    plotCorrelation_option = ' --corMethod spearman --whatToPlot heatmap --plotNumbers '
    if 'plotCorrelation_option' in configDict:
        plotCorrelation_option = configDict['plotCorrelation_option']

    plotPCA_option = ''
    if 'plotPCA_option' in configDict:
        plotPCA_option = configDict['plotPCA_option']

    bam_to_process = '.bam '.join(bam_list)
    bam_to_process = bam_to_process + '.bam'
    command_bamSum = 'multiBamSummary ' + multiBamSummary_option + ' --
bamfiles ' + bam_to_process +' -out ' + mutliBamSum_dir + '/multiBam_results.npz --
outRawCounts  ' + mutliBamSum_dir + '/multiBam_rawCount.csv\n'
    command_bamCor = 'plotCorrelation ' + plotCorrelation_option + ' --
corData ' + mutliBamSum_dir + '/multiBam_results.npz --
plotFile ' + mutliBamSum_dir + '/correlation_graph.pdf --
outFileCorMatrix ' + mutliBamSum_dir + '/correlation_score.csv\n'
    command_bamPCA = 'plotPCA ' + plotPCA_option + ' --
corData ' + mutliBamSum_dir + '/multiBam_results.npz --
plotFile ' + mutliBamSum_dir + '/PCA_graph.pdf --
outFileNameData ' + mutliBamSum_dir + '/PCA_matrix.csv \n'
    # tmpdir variable
    tmpdir = 'TMPDIR_' + job_basename
    # Write and execute the script
    script = open(scriptDir + 'multiBamSummary.sh', 'w')
    script.write('#!/bin/bash\n#$ -m a\n#$ -q workq\n')
    script.write('#$ -l h_vmem=20G\n#$ -l mem=20G\n')
    script.write('#$ -o ' + logDir + 'multiBamSummary.out\n')
    script.write('#$ -e ' + logDir + 'multiBamSummary.err\n')
    script.write('#$ -N ' + job_basename + '\n')
    script.write(wait_condition)
    script.write(tmpdir + '=$(mktemp -d /tmp/multiBamSum_XXXXXX) \n')
    script.write('cd $' + tmpdir + ' \n')
    for bam in bam_to_process.split(' '):
        script.write('ln -s ' + bamDir + '/' + bam + '* .\n')
    script.write(command_bamSum)
    script.write(command_bamCor)
    script.write(command_bamPCA)
    script.write('rm -r $' + tmpdir)
    script.write('\n#end of the script\n')
    script.close()
```

```
68.     os.system('qsub ' + scriptDir + 'multiBamSummary.sh')
69. 
70. 
71. ##############
72. ####
73. ## CALL
74. #
75. def multiBamSummary(configDict):
76.     print('multiBamSummary project : ' + configDict['project'])
77.     #Call function to run command
78.     multiBamSummary_Commands(configDict)
79.     return(configDict)
```

runFastqToMap_multiBamSummary.py

```
1.  #!/usr/bin/env python3
2.  # Author: Ismael padioleau
3.  # Contact: ismael.padioleau@igh.cnrs.fr
4.  # April 10th 2017
5.  # Run multi bam summary from deeptools
6.  
7.  import os
8.  
9.  ####
10. ## FUNCTIONS
11. #
12. def multiBamSummary_Commands(configDict):
13.     #set variables
14.     bam_list = configDict['bam_list']
15.     uid = configDict['uid']
16.     project = configDict['project']
17.     bamDir = configDict['bam_dir']
18.     mutliBamSum_dir = configDict['mutliBamSum_dir']
19.     scriptDir = mutliBamSum_dir + '/script/'
20.     logDir = mutliBamSum_dir + '/log/'
21.     job_basename = 'job_' + uid + '_deeptool_multiBamSummary'
22.     wait_condition = '#\n'
23.     if 't1.1_run' in configDict or 't1.2_run' in configDict:
24.         wait_condition = '#$ -hold_jid job_' + uid + '_fastqToBam_samToBam_* \n'
25.     #build sam directory
26.     if not os.path.exists(logDir):
27.        os.makedirs(logDir)
28.     if not os.path.exists(scriptDir):
29.        os.makedirs(scriptDir)
30.     #create commands
31.     multiBamSummary_option = ' -bins -binSize 10000 '
32.     if 'multiBamSummary_option' in configDict:
33.         multiBamSummary_option = configDict['multiBamSummary_option']
34. 
35.     plotCorrelation_option = ' -corMethod spearman -whatToPlot heatmap '
36.     if 'plotCorrelation_option' in configDict:
37.         plotCorrelation_option = configDict['plotCorrelation_option']
38. 
39.     plotPCA_option = ''
40.     if 'plotPCA_option' in configDict:
41.         plotPCA_option = configDict['plotPCA_option']
42. 
43.     bam_to_process = '.bam '.join(bam_list)
44.     command_bamSum = 'multiBamSummary ' + multiBamSummary_option + ' -
    bams ' + bam_to_process +' -out ' + mutliBamSum_dir + '/multiBam_results.npz\n'
45.     command_bamCor = 'plotCorrelation ' + plotCorrelation_option + ' -
    corData ' + mutliBamSum_dir + '/multiBam_results.npz -
    plotFile ' + mutliBamSum_dir + '/correlation_graph.pdf\n'
```

```
46.     command_bamPCA = 'plotPCA ' + plotPCA_option + ' -
    corData ' + mutliBamSum_dir + '/multiBam_results.npz -
    plotFile ' + mutliBamSum_dir + '/PCA_graph.pdf\n'
47.     # Write and execute the script
48.     script = open(scriptDir + basename + '_fastqToBam_bamToBed_.sh', 'w')
49.     script.write('#!/bin/bash\n#$ -m a\n#$ -q workq\n')
50.     script.write('#$ -o ' + logDir + 'multiBamSummary.out\n')
51.     script.write('#$ -e ' + logDir + 'multiBamSummary.err\n')
52.     script.write('#$ -N ' + job_basename + '\n')
53.     script.write(wait_condition)
54.     script.write(command_bamSum)
55.     script.write(command_bamCor)
56.     script.write(command_bamPCA)
57.     script.write('\n#end of the script\n')
58.     script.close()
59.     os.system('qsub ' + scriptDir + basename + '_fastqToBam_bamToBed_.sh')
60.
61.
62. ##############
63. ####
64. ## CALL
65. #
66. def multiBamSummary(configDict):
67.     print('multiBamSummary project : ' + configDict['project'])
68.     #Call function to run command
69.     multiBamSummary_Commands(configDict)
70.     return(configDict)
```

runFastqToMap_bedtools.py

```
1.  #!/usr/bin/env python3
2.  # Author: Ismael padioleau
3.  # Contact: ismael.padioleau@igh.cnrs.fr
4.  # May 19th 2014
5.  # Manage the pipeline from fastq to mapped reads
6.  # Function exported fastqTomap
7.
8.  import os
9.
10. ####
11. ## FUNCTIONS
12. #
13. def bamToBed_Commands(basename, configDict):
14.     #set variables
15.     bedtools = configDict['bedtools_dir']
16.     uid = configDict['uid']
17.     project = configDict['project']
18.     bamDir = configDict['bam_dir']
19.     bedDir = configDict['bed_dir']
20.     scriptDir = bedDir + '/script/'
21.     logDir = bedDir + '/log/'
22.     job_basename = 'job_' + uid + '_fastqToBam_bamToBed_' + basename
23.     wait_condition = '#\n'
24.     if 't1.1_run' in configDict or 't1.2_run' in configDict:
25.         wait_condition = '#$ -hold_jid job_' + uid + '_fastqToBam_samToBam_* \n'
26.     #build sam directory
27.     if not os.path.exists(logDir):
28.        os.makedirs(logDir)
29.     if not os.path.exists(scriptDir):
30.        os.makedirs(scriptDir)
31.     #create commands
32.     command_bam = bedtools  + ' bamtobed -
    i ' + bamDir + '/' + basename + '.bam > ' + bedDir + '/' + basename + '.bed\n'
33.     # Write and execute the script
```

```
    script = open(scriptDir + basename + '_fastqToBam_bamToBed_.sh', 'w')
    script.write('#!/bin/bash\n#$ -m a\n#$ -q workq\n')
    script.write('#$ -o ' + logDir + basename + '_bamToBed.out\n')
    script.write('#$ -e ' + logDir + basename + '_bamToBed.err\n')
    script.write('#$ -N ' + job_basename + '\n')
    script.write(wait_condition)
    script.write(command_bam)
    script.write('\n#end of the script\n')
    script.close()
    os.system('qsub ' + scriptDir + basename + '_fastqToBam_bamToBed_.sh')


##############
####
## CALL
#
def bamToBed(configDict):
    print('bamToBed project : ' + configDict['project'])
    #Call function to run command for each sample
    for name in  configDict['bam_list']:
        bamToBed_Commands(name, configDict)
    return(configDict)
```

runFastqToMap_bowtie.py

```
#!/usr/bin/env python3
# Author: Ismael padioleau
# Contact: ismael.padioleau@igh.cnrs.fr
# May 15th 2014
# Manage the pipeline from fastq to mapped reads
# Function exported fastqTomap

import os

def fastqToSam_Commands(basename, configDict):
    #set variables
    bowtie = configDict['bowtie_dir']
    samtools = configDict['samtools']
    ref_genome = configDict['ref_genome_name']
    uid = configDict['uid']
    project = configDict['project']
    bamDir = configDict['bam_dir']
    fastqDir = configDict['fastq_dir']
    logDir = bamDir + '/log/'
    scriptDir = bamDir + '/script/'
    job_basename_bam = 'job_' + uid + '_fastqToBam_samToBam_'
    queue = 'workq'
    #build directories
    if not os.path.exists(logDir):
       os.makedirs(logDir)
    if not os.path.exists(scriptDir):
       os.makedirs(scriptDir)
    #Check bowtie commands and options
    if ('bowtie' not in configDict or len(configDict['bowtie']) == 0):
        bowtieOption = ' -p 8 -x '
    else:
        bowtieOption = ' -p 8 ' + configDict['bowtie'] + ' -x '
    #create and send commands
    if configDict['pairend'] == '1':
        command_bam = bowtie + bowtieOption + ref_genome + ' -
1 ' + fastqDir + '/' + basename + '_1.fastq.gz -
2 ' + fastqDir + '/' + basename + '_2.fastq.gz | ' + samtools + ' view -Sb -
 | ' + samtools + ' sort - -
```

```
    o ' + bamDir + '/' + basename + '.bam ; ' + samtools + ' index ' + bamDir + '/' + basename + '
    .bam'
36.     else:
37.         command_bam = bowtie + bowtieOption + ref_genome + ' -
    U ' + fastqDir + '/' + basename + '_1.fastq.gz | ' + samtools + ' view -Sb -
     | ' + samtools + ' sort - -
    o ' + bamDir + '/' + basename + '.bam ; ' + samtools + ' index ' + bamDir + '/' + basename + '
    .bam'
38.     # Write and execute the script
39.     script = open(scriptDir + basename + '_bowtie2.sh', 'w')
40.     script.write('#!/bin/bash\n#$ -m a\n#$ -q workq\n')
41.     script.write('#$ -o ' + logDir + basename + '.out\n')
42.     script.write('#$ -e ' + logDir + basename + '.err\n')
43.     script.write('#$ -l h_vmem=5G\n#$ -l mem=5G\n#$ -pe parallel_smp 8\n')
44.     script.write('#$ -N ' + job_basename_bam + basename + '\n')
45.     script.write(command_bam)
46.     script.write('\n#end of the script\n')
47.     script.close()
48.     os.system('qsub ' + scriptDir + basename + '_bowtie2.sh')
49.
50. ##############
51.
52. def fastqToBam_bowtie(configDict):
53.     print('fastqToSam project : ' + configDict['project'])
54.     #Call function to run command for each sample
55.     for name in  configDict['fastq_list']:
56.         fastqToSam_Commands(name, configDict)
57.     return(configDict)
```

runFastqToMap_deepTools.py

```
1.  #!/usr/bin/env python3
2.  # Author: Ismael padioleau
3.  # Contact: ismael.padioleau@igh.cnrs.fr
4.  # May 19th 2014
5.  # Manage the pipeline from fastq to mapped reads
6.  # Function exported fastqTomap
7.
8.  import os
9.
10. ####
11. ## FUNCTIONS
12. #
13. def bamToBigWig_Commands(basename, configDict):
14.     #set variables
15.     uid = configDict['uid']
16.     project = configDict['project']
17.     profileDir = configDict['profileDir']
18.     bamDir = configDict['bam_dir']
19.     scriptDir = profileDir + '/script/'
20.     logDir = profileDir + '/log/'
21.     wait_condition = '#\n'
22.     extension='.bw'
23.     job_basename = 'job_' + uid + '_bamToBigWig' + basename
24.     if 'binSize_bigwig' in configDict:
25.         binSize = configDict['binSize_bigwig'] + ' '
26.     else:
27.         binSize = '10 '
28.     if 'fragmentLength' in configDict:
29.         fragmentLength = ' -e ' + configDict['fragmentLength']
30.     else:
31.         fragmentLength = ''
32.     if 'deeptools_format' in configDict:
33.         deeptools_format = ' --outFileFormat ' + configDict['deeptools_format'] + ' '
```

```
        if configDict['deeptools_format'] == 'bedgraph':
            extension='.bdg'
    else:
        deeptools_format = ''
    #Set waitcondition
    if 't1.1_run' in configDict or 't1.2_run' in configDict:
        wait_condition = '#$ -hold_jid job_' + uid + '_fastqToBam_samToBam_* \n'
    #build sam directory
    if not os.path.exists(logDir):
       os.makedirs(logDir)
    if not os.path.exists(scriptDir):
       os.makedirs(scriptDir)
    #create commands
    command_bam = 'bamCoverage -p 8 --bam ' + bamDir + '/' + basename + '.bam --
binSize ' + binSize + fragmentLength + deeptools_format + ' -
o ' + profileDir + '/' + basename + extension
    # Write and execute the script
    script = open(scriptDir + basename + '_deeptools_bamToBigWig.sh', 'w')
    script.write('#!/bin/bash\n#$ -m a\n#$ -q workq\n')
    script.write('#$ -l h_vmem=80G\n#$ -l mem=80G\n')
    script.write('#$ -o ' + logDir + basename + '_bamToBigWig.out\n')
    script.write('#$ -e ' + logDir + basename + '_bamToBigWig.err\n')
    script.write('#$ -N ' + job_basename + '\n')
    script.write(wait_condition)
    script.write(command_bam)
    script.write('\n#end of the script\n')
    script.close()
    os.system('qsub ' + scriptDir + basename + '_deeptools_bamToBigWig.sh')


def bamRatio_Commands(basename, configDict):
    #set variables
    uid = configDict['uid']
    bamDir = configDict['bam_dir']
    ratio_dir = configDict['ratio_dir']
    logDir = ratio_dir + '/log/'
    scriptDir = ratio_dir + '/script/'
    job_basename_ratio = 'job_' + uid + '_ratio_'
    queue = 'workq'
    pair_name = basename[0].split('.bam')[0] + '_' + basename[1].split('.bam')[0]
    wait_condition = '#\n'
    extension='.bw'
    if 't1.1_run' in configDict or 't1.2_run' in configDict:
        wait_condition = '#$ -hold_jid job_' + uid + '_fastqToBam_samToBam_* \n'
    #build directories
    if not os.path.exists(logDir):
       os.makedirs(logDir)
    if not os.path.exists(scriptDir):
       os.makedirs(scriptDir)
    if 'binSize_bigwig' in configDict:
        binSize = ' --binSize ' + configDict['binSize_bigwig']
    else:
        binSize = ' --binSize 10'
    if 'fragmentLength' in configDict:
        fragmentLength = ' -e ' + configDict['fragmentLength']
    else:
        fragmentLength = ''
    if 'deeptools_format' in configDict:
        deeptools_format = ' --outFileFormat ' + configDict['deeptools_format'] + ' '
        if configDict['deeptools_format'] == 'bedgraph':
            extension='.bdg'
    else:
        deeptools_format = ''
    # Create a unique var name to save temporary directory
    pair_name = pair_name.replace('-','_')
    pair_name = pair_name.replace('.','_')
```

```
    pair_name = pair_name.replace(' ','_')
    tmpdir = 'TMPDIR_' + pair_name
    #create and send commands
    command_ratio = 'bamCompare -p 8 --bamfile1 ' + bamDir + '/' + basename[0] + ' --
bamfile2 ' + bamDir + '/' + basename[1] + binSize + fragmentLength + deeptools_format + ' --
ratio ratio -o ' + pair_name + extension + '\n'
# Write and execute the script
    script = open(scriptDir + pair_name + '_ratio.sh', 'w')
    script.write('#!/bin/bash\n#$ -m a\n#$ -q workq\n')
    script.write('#$ -o ' + logDir + pair_name + '.out\n')
    script.write('#$ -e ' + logDir + pair_name + '.err\n')
    script.write('#$ -l h_vmem=100G\n#$ -l mem=100G\n')
    script.write('#$ -N ' + job_basename_ratio + pair_name + '\n')
    script.write(wait_condition)
    script.write(tmpdir + '=$(mktemp -d /tmp/ratio_XXXXXX) \n')
    script.write('cd $' + tmpdir + ' \n')
    script.write(command_ratio)
    script.write('mv $' + tmpdir + '/* ' + ratio_dir + '\n')
    script.write('#rmdir $' + tmpdir)
    script.write('\n#end of the script\n')
    script.close()
    os.system('qsub ' + scriptDir + pair_name + '_ratio.sh')

##############
####
## CALL
#
def bamToBigWig(configDict):
    print('BamToBigWig project : ' + configDict['project'])
    #Call function to run command for each sample
    for name in  configDict['bam_list']:
        bamToBigWig_Commands(name, configDict)
    return(configDict)

def bamRatio(configDict):
    print('BamRatio project : ' + configDict['project'])
    #Call function to run command for each sample
    for pair in  configDict['pair_list']:
        bamRatio_Commands(pair, configDict)
    return(configDict)
```

runFastqToMap_median_center.py

```
#!/usr/bin/env python3
# Author: Ismael padioleau
# Contact: ismael.padioleau@igh.cnrs.fr
# May 19th 2014
# Manage the pipeline from fastq to mapped reads
# Function exported fastqTomap

import os

####
## FUNCTIONS
#
def bamToBigWig_Commands(basename, configDict):
    #set variables
    uid = configDict['uid']
    project = configDict['project']
    bigWigDir = configDict['bigWigDir']
    bamDir = configDict['bam_dir']
    scriptDir = bigWigDir + '/script/'
    logDir = bigWigDir + '/log/'
    wait_condition = '#\n'
```

```
    job_basename = 'job_' + uid + '_bamToBigWig' + basename
    if 'binSize_bigwig' in configDict:
        binSize = configDict['binSize_bigwig']
    else:
        binSize = '10'
    #Set waitcondition
    if 't1.1_run' in configDict or 't1.2_run' in configDict:
        wait_condition = '#$ -hold_jid job_' + uid + '_fastqToBam_samToBam_* \n'
    #build sam directory
    if not os.path.exists(logDir):
       os.makedirs(logDir)
    if not os.path.exists(scriptDir):
       os.makedirs(scriptDir)
    #create commands
    command_bam = 'bamCoverage --bam ' + bamDir + '/' + basename + '.bam --
binSize ' + binSize + ' -o ' + bigWigDir + '/' + basename + '.bw'
    # Write and execute the script
    script = open(scriptDir + basename + '_deeptools_bamToBigWig.sh', 'w')
    script.write('#!/bin/bash\n#$ -m a\n#$ -q workq\n')
    script.write('#$ -l h_vmem=15G\n#$ -l mem=15G\n')
    script.write('#$ -o ' + logDir + basename + '_bamToBigWig.out\n')
    script.write('#$ -e ' + logDir + basename + '_bamToBigWig.err\n')
    script.write('#$ -N ' + job_basename + '\n')
    script.write(wait_condition)
    script.write(command_bam)
    script.write('\n#end of the script\n')
    script.close()
    os.system('qsub ' + scriptDir + basename + '_deeptools_bamToBigWig.sh')


def bamRatio_Commands(basename, configDict):
    #set variables
    uid = configDict['uid']
    bamDir = configDict['bam_dir']
    ratio_dir = configDict['ratio_dir']
    logDir = ratio_dir + '/log/'
    scriptDir = ratio_dir + '/script/'
    job_basename_ratio = 'job_' + uid + '_ratio_'
    queue = 'workq'
    pair_name = basename[0].split('.bam')[0] + '_' + basename[1].split('.bam')[0]
    wait_condition = '#\n'
    if 't1.1_run' in configDict or 't1.2_run' in configDict:
        wait_condition = '#$ -hold_jid job_' + uid + '_fastqToBam_samToBam_* \n'
    #build directories
    if not os.path.exists(logDir):
       os.makedirs(logDir)
    if not os.path.exists(scriptDir):
       os.makedirs(scriptDir)
    # Create a unique var name to save temporary directory
    pair_name = pair_name.replace('-','_')
    pair_name = pair_name.replace('.','_')
    pair_name = pair_name.replace(' ','_')
    tmpdir = 'TMPDIR_' + pair_name
    #create and send commands
    command_ratio = 'bamCompare --bamfile1 ' + bamDir + '/' + basename[0] + ' --
bamfile2 ' + bamDir + '/' + basename[1] + ' -bs 1 --ratio ratio --outFileFormat bigwig -
o ' + pair_name + '.bw\n'
# Write and execute the script
    script = open(scriptDir + pair_name + '_ratio.sh', 'w')
    script.write('#!/bin/bash\n#$ -m a\n#$ -q workq\n')
    script.write('#$ -o ' + logDir + pair_name + '.out\n')
    script.write('#$ -e ' + logDir + pair_name + '.err\n')
    script.write('#$ -l h_vmem=20G\n#$ -l mem=20G\n')
    script.write('#$ -N ' + job_basename_ratio + pair_name + '\n')
    script.write(wait_condition)
    script.write(tmpdir + '=$(mktemp -d /tmp/ratio_XXXXXX) \n')
```

```
    script.write('cd $' + tmpdir + ' \n')
    script.write(command_ratio)
    script.write('mv $' + tmpdir + '/* ' + ratio_dir + '\n')
    script.write('#rmdir $' + tmpdir)
    script.write('\n#end of the script\n')
    script.close()
    os.system('qsub ' + scriptDir + pair_name + '_ratio.sh')

##############
####
## CALL
#
def bamToBigWig(configDict):
    print('BamToBigWig project : ' + configDict['project'])
    #Call function to run command for each sample
    for name in  configDict['bam_list']:
        bamToBigWig_Commands(name, configDict)
    return(configDict)

def bamRatio(configDict):
    print('BamRatio project : ' + configDict['project'])
    #Call function to run command for each sample
    for pair in  configDict['pair_list']:
        bamRatio_Commands(pair, configDict)
    return(configDict)
```

runFastqToMap_samtools.py

```
#!/usr/bin/env python3
# Author: Ismael padioleau
# Contact: ismael.padioleau@igh.cnrs.fr
# May 19th 2014
# Manage the pipeline from fastq to mapped reads
# Function exported fastqTomap

import os

####
## FUNCTIONS
#
def samToBam_Commands(basename, configDict):
    #set variables
    samtools = configDict['samtools']
    uid = configDict['uid']
    project = configDict['project']
    samDir = configDict['sam_dir']
    bamDir = configDict['bam_dir']
    scriptDir = bamDir + '/script/'
    logDir = bamDir + '/log/'
    job_basename = 'job_' + uid + '_fastqToBam_samToBam_' + basename
    #build sam directory
    if not os.path.exists(logDir):
       os.makedirs(logDir)
    if not os.path.exists(scriptDir):
       os.makedirs(scriptDir)
    #create commands
    command_bam =  samtools + ' view -
Sb ' + samDir + '/' + basename + '.sam | ' + samtools + ' sort -
 ' + bamDir + '/' + basename + '.bam; ' + samtools + ' index ' + bamDir + '/' + basename + '.b
am'

    # Write and execute the script
    script = open(scriptDir + basename + '_samtools_samToBam.sh', 'w')
    script.write('#!/bin/bash\n#$ -m a\n#$ -q workq\n')
```

```
34.     script.write('#$ -o ' + logDir + basename + '_samToBam.out\n')
35.     script.write('#$ -e ' + logDir + basename + '_samToBam.err\n')
36.     script.write('#$ -N ' + job_basename + '\n')
37.     script.write(command_bam)
38.     script.write('\n#end of the script\n')
39.     script.close()
40.     os.system('qsub ' + scriptDir + basename + '_samtools_samToBam.sh')
41. 
42. 
43. def sortBam_Commands(basename, configDict):
44.     #set variables
45.     samtools = configDict['samtools']
46.     uid = configDict['uid']
47.     project = configDict['project']
48.     bamDir = configDict['bam_dir']
49.     outDir = configDict['sort_bam_dir']
50.     scriptDir = outDir + '/script/'
51.     logDir = outDir + '/log/'
52.     wait_condition = ' '
53.     job_basename = 'job_' + uid + '_fastqToBam_SortBam_' + basename
54.     #Set wait condition
55.     if 't1.2_run' in configDict:
56.         info = basename.split('_')
57.         lane = info[len(info)-1]
58.         sample_id = '_'.join(info[0:len(info)-2])
59.         wait_condition = '#$ -
    hold_jid job_' + uid + '_fastqToBam_samToBam_' + basename + ' \n'
60.     #build sam directory
61.     if not os.path.exists(logDir):
62.        os.makedirs(logDir)
63.     if not os.path.exists(scriptDir):
64.        os.makedirs(scriptDir)
65.     #create and send commands
66.     command =  samtools + ' sort ' + bamDir + '/' + basename + '.bam ' + outDir + '/' + basena
    me
67. 
68.     # Write and execute the script
69.     script = open(scriptDir + basename + '_samtools_sort_bam.sh', 'w')
70.     script.write('#!/bin/bash\n#$ -m a\n#$ -q workq\n')
71.     script.write('#$ -o ' + logDir + basename + '_samtools_sort_bam.out\n')
72.     script.write('#$ -e ' + logDir + basename + '_samtools_sort_bam.err\n')
73.     script.write('#$ -N ' + job_basename + '\n')
74.     script.write(wait_condition)
75.     script.write(command)
76.     script.write('\n#end of the script\n')
77.     script.close()
78.     os.system('qsub ' + scriptDir + basename + '_samtools_sort_bam.sh')
79. 
80. 
81. def indexBam_Commands(basename, configDict):
82.     #set variables
83.     samtools = configDict['samtools']
84.     uid = configDict['uid']
85.     project = configDict['project']
86.     bamDir = configDict['bam_dir']
87.     outDir = configDict['index_bam_dir']
88.     scriptDir = outDir + '/script/'
89.     logDir = outDir + '/log/'
90.     wait_condition = '#\n'
91.     job_basename = 'job_' + uid + '_fastqToBam_IndexBam_' + basename
92.     info = basename.split('_')
93.     lane = info[len(info)-1]
94.     sample_id = '_'.join(info[0:len(info)-2])
95.     if 't1.2_run' in configDict:
96.         wait_condition = '#$ -
    hold_jid job_' + uid + '_fastqToBam_SortBam_' + basename + ' \n'
```

```
97.         bamDir = configDict['sort_bam_dir']
98.     #build sam directory
99.     if not os.path.exists(logDir):
100.        os.makedirs(logDir)
101.    if not os.path.exists(scriptDir):
102.       os.makedirs(scriptDir)
103.    #create and send commands
104.    command =  samtools + ' index ' + bamDir + '/' + basename + '.bam ' + outDir + '/' + basen
    ame + '.bam.bai'
105.    if 'multiple_index_bam_dir' in configDict:
106.        if os.path.isfile(bamDir + '/' + basename + '.bam'):
107.            # Write and execute the script
108.            script = open(scriptDir + basename + '_samtools_index_bam.sh', 'w')
109.            script.write('#!/bin/bash\n#$ -m a\n#$ -q workq\n')
110.            script.write('#$ -o ' + logDir + basename + '_samtools_index_bam.out\n')
111.            script.write('#$ -e ' + logDir + basename + '_samtools_index_bam.err\n')
112.            script.write('#$ -N ' + job_basename + '\n')
113.            script.write(wait_condition)
114.            script.write(command)
115.            script.write('\n#end of the script\n')
116.            script.close()
117.            os.system('qsub ' + scriptDir + basename + '_samtools_index_bam.sh')
118.    else:
119.        # Write and execute the script
120.        script = open(scriptDir + basename + '_samtools_index_bam.sh', 'w')
121.        script.write('#!/bin/bash\n#$ -m a\n#$ -q workq\n')
122.        script.write('#$ -o ' + logDir + basename + '_samtools_index_bam.out\n')
123.        script.write('#$ -e ' + logDir + basename + '_samtools_index_bam.err\n')
124.        script.write('#$ -N ' + job_basename + '\n')
125.        script.write(wait_condition)
126.        script.write(command)
127.        script.write('\n#end of the script\n')
128.        script.close()
129.        os.system('qsub ' + scriptDir + basename + '_samtools_index_bam.sh')
130.
131.
132.##############
133.####
134.## CALL
135.#
136.def samToBam(configDict):
137.    print('samToBam project : ' + configDict['project'])
138.    #Call function to run command for each sample
139.    for name in  configDict['sam_list']:
140.        samToBam_Commands(name, configDict)
141.    return(configDict)
142.
143.def sortBam(configDict):
144.    print('SortBam project : ' + configDict['project'])
145.    #Call function for each samples
146.    for name in configDict['bam_list']:
147.        sortBam_Commands(name,configDict)
148.    return(configDict)
149.
150.def indexBam(configDict):
151.    print('IndexBam project : ' + configDict['project'])
152.    #Call function for each samples
153.    for name in configDict['bam_list']:
154.        indexBam_Commands(name,configDict)
155.    return(configDict)
```

runPeakCalling_macs.py

```
1.  #!/usr/bin/env python3
```

```
2.  # Author: Ismael padioleau
3.  # Contact: ismael.padioleau@igh.cnrs.fr
4.  # June 2sd 2014
5.  # Run macs
6.
7.  import os
8.
9.  def peakCalling_macs_Commands(basename, configDict):
10.     #set variables
11.     macs = configDict['macs_dir']
12.     ref_genome = configDict['ref_genome_name']
13.     uid = configDict['uid']
14.     project = configDict['project']
15.     bamDir = configDict['bam_dir']
16.     pic_dir = configDict['peak_dir']
17.     logDir = pic_dir + '/log/'
18.     scriptDir = pic_dir + '/script/'
19.     job_basename_macs = 'job_' + uid + '_peakCalling_'
20.     queue = 'workq'
21.     pair_name = basename[0].split('.bam')[0] + '_' + basename[1].split('.bam')[0]
22.     wait_condition = '#\n'
23.     if 't1.1_run' in configDict or 't1.2_run' in configDict:
24.         wait_condition = '#$ -hold_jid job_' + uid + '_fastqToBam_samToBam_* \n'
25.     #build directories
26.     if not os.path.exists(logDir):
27.        os.makedirs(logDir)
28.     if not os.path.exists(scriptDir):
29.        os.makedirs(scriptDir)
30.     #Check macs commands and options
31.     macsOption = configDict['macs_option']
32.     # Create a unique var name to save temporary directory
33.     pair_name = pair_name.replace('-','_')
34.     pair_name = pair_name.replace('.','_')
35.     pair_name = pair_name.replace(' ','_')
36.     tmpdir = 'TMPDIR_' + pair_name
37.     #create and send commands
38.     command_macs = macs + ' callpeak ' + macsOption + ' -
    t ' + bamDir + '/' + basename[0] + ' -c ' + bamDir + '/' + basename[1] + ' -
    n ' + pair_name + '\n'
39. # Write and execute the script
40.     script = open(scriptDir + pair_name + '_macs.sh', 'w')
41.     script.write('#!/bin/bash\n#$ -m a\n#$ -q workq\n')
42.     script.write('#$ -o ' + logDir + pair_name + '.out\n')
43.     script.write('#$ -e ' + logDir + pair_name + '.err\n')
44.     script.write('#$ -l h_vmem=10G\n#$ -l mem=10G\n')
45.     script.write('#$ -N ' + job_basename_macs + pair_name + '\n')
46.     script.write(wait_condition)
47.     script.write(tmpdir + '=$(mktemp -d /tmp/macs_XXXXXX) \n')
48.     script.write('cd $' + tmpdir + ' \n')
49.     script.write(command_macs)
50.     script.write('mv $' + tmpdir + '/* ' + pic_dir + '\n')
51.     script.write('#rmdir $' + tmpdir)
52.     script.write('\n#end of the script\n')
53.     script.close()
54.     os.system('qsub ' + scriptDir + pair_name + '_macs.sh')
55.
56. ##############
57.
58. def peakCalling_macs(configDict):
59.     print('Peak calling on project : ' + configDict['project'])
60.     #Call function to run command for each sample
61.     for pair in  configDict['pair_list']:
62.         peakCalling_macs_Commands(pair, configDict)
63.     return(configDict)
```

runPeakCalling_sicer.py

```python
#!/usr/bin/env python3
# Author: Ismael padioleau
# Contact: ismael.padioleau@igh.cnrs.fr
# June 2sd 2014
# Run sicer on a list of pair IP/Input

import os

def peakCalling_sicer_Commands(basename, configDict):
    #set variables
    sicer = configDict['sicer_dir']
    ref_genome = configDict['ref_genome_name']
    uid = configDict['uid']
    project = configDict['project']
    bedDir = configDict['bed_dir']
    pic_dir = configDict['peak_dir']
    logDir = pic_dir + '/log/'
    scriptDir = pic_dir + '/script/'
    job_basename_sicer = 'job_' + uid + '_peakCalling_'
    queue = 'workq'
    pair_name = basename[0].split('.bed')[0] + '_' + basename[1].split('.bed')[0]
    wait_condition = '#\n'
    if 't1.1_run' in configDict or 't1.2_run' in configDict:
        wait_condition = '#$ -hold_jid job_' + uid + '_fastqToBam_samToBam_* \n'
    #build directories
    if not os.path.exists(logDir):
       os.makedirs(logDir)
    if not os.path.exists(scriptDir):
       os.makedirs(scriptDir)
    #Check sicer commands and options
    sicerOption = configDict['sicer_option']
    # Create a unique var name to save temporary directory
    pair_name = pair_name.replace('-','_')
    pair_name = pair_name.replace('.','_')
    pair_name = pair_name.replace(' ','_')
    tmpdir = 'TMPDIR_' + pair_name
# Write and execute the script
    script = open(scriptDir + pair_name + '_sicer.sh', 'w')
    script.write('#!/bin/bash\n#$ -m a\n#$ -q workq\n')
    script.write('#$ -o ' + logDir + pair_name + '.out\n')
    script.write('#$ -e ' + logDir + pair_name + '.err\n')
    script.write('#$ -l h_vmem=10G\n#$ -l mem=10G\n')
    script.write('#$ -N ' + job_basename_sicer + pair_name + '\n')
    script.write(wait_condition)
    script.write(tmpdir + '=$(mktemp -d /tmp/sicer_XXXXXX) \n')
    script.write('cd $' + tmpdir + ' \n')
    script.write(sicer + ' ' + bedDir + ' ' + basename[0] + ' ' + basename[1] + ' . ' +  sicer
Option + '\n')
    script.write('mv $' + tmpdir + '/* ' + pic_dir + '\n')
    script.write('rmdir $' + tmpdir)
    script.write('\n#end of the script\n')
    script.close()
    os.system('qsub ' + scriptDir + pair_name + '_sicer.sh')

##############

def peakCalling_sicer(configDict):
    print('Peak calling on project : ' + configDict['project'])
    #Call function to run command for each sample
    for pair in  configDict['pair_list']:
        peakCalling_sicer_Commands(pair, configDict)
    return(configDict)
```

configParser.py

```
#!/usr/bin/env python
#Author: Ismael Padioleau
# May 15th 2014
#parse the config file
#functions exported: getConfigDict

from sys import stderr

def getConfigDict(configFilePath):
    #read in the config
    configFileHandle = open(configFilePath,'r')
    configDict = dict()
    for line in configFileHandle:
        # ignore empty and comment lines
        if line[0] == '#' or not line.strip():
            continue
        configList = (line.rstrip('\n')).split(',')
        configDict[configList[0]] = " ".join(configList[1:])
    configFileHandle.close()
    #print(configDict)
    #check that specified
    return configDict
```

distance_between_reads.py

```
#!/usr/bin/python2.7
# Author Ismael padioleau
# Read a sorted sam file from stdin (samtools view example.bam | python2.7 distance_between_reads.py)
# Check where is mapped the next read and keep results in a list (0 if overlapped)
# Output in tsv format:
# reference distance    count

# Import
import sys
import os

# Args
out = open(os.path.abspath(sys.argv[1]),'a')
quality_threshold = int(sys.argv[2])

# functions
def found_first_valid_read( line, qual, quality_threshold, previous_ref):
    #print(line)
    previous_pos = 0
    if not line[0] == '@':
        qual = int(line.split('\t')[4])
        if(qual >= quality_threshold):
            previous_pos = int(line.split('\t')[3])
            previous_ref = line.split('\t')[2]
    return previous_pos, previous_ref, qual

# Prepare with first valid read
qual = -1
previous_ref = ''
while(qual < quality_threshold):
    previous_pos, previous_ref, qual = found_first_valid_read(sys.stdin.readline(), qual, quality_threshold, previous_ref)

```

```
count = {} # Keep count per references (chromosomes)
total_count = {} # Keep count for the whole genome
total_reads = 0.0

# Process all reads
for line in sys.stdin:
    pos = int(line.split('\t')[3])
    ref = line.split('\t')[2]
    qual = int(line.split('\t')[4])
    if not ref == previous_ref:
        qual = -1
        previous_pos, previous_ref, qual = found_first_valid_read(line, qual, quality_threshold, previous_ref)
    else:
        if qual < quality_threshold:
            continue
        if not ref in count:
            count[ref] = {}
        dist = pos - previous_pos
        previous_pos = pos
        if dist in count[ref]:
            total_reads += 1
            new_count = count[ref][dist] + 1
            new_count_total = total_count[dist] + 1
            count[ref][dist] = new_count
            total_count[dist] = new_count_total
        else:
            total_reads += 1
            count[ref][dist] = 1
            if dist in total_count:
                total_count[dist] += 1
            else:
                total_count[dist] = 1

# Write report
for ref in count.keys():
    for dist in count[ref].keys():
        out.write(ref + '\t' + str(dist) + '\t' + str((count[ref][dist]/total_reads)*100).replace('.',',') + '\n')

for dist in total_count.keys():
    out.write('genome\t' + str(dist) + '\t' + str((total_count[dist]/total_reads)*100).replace('.',',') + '\n')

out.close()
```

distance_between_reads_summary.py

```
#!/usr/local/bioinfo/src/python/current/bin/python

import os
import sys
import fnmatch


inputDir = os.path.abspath(sys.argv[1])
output = open(os.path.abspath(sys.argv[2]), 'a')
upperlimit = int(sys.argv[3]) + 1

summary_dict = {}
name_list = []
for file1 in os.listdir(inputDir):
    if fnmatch.fnmatch(file1, '*' + 'distance_between_reads.csv'):
        name = file1.split('_distance_between_reads.csv')[0]
```

```
        name_list.append(name)

for i in range(0,upperlimit):
    summary_dict[i] = []

for name in name_list:
    FILE1 = open(inputDir + '/' + name + '_distance_between_reads.csv', 'r')
    tmp_dict = {}
    for line in FILE1:
        info = line.strip().split('\t')
        if int(info[1]) <= upperlimit and info[0] == 'genome':
            tmp_dict[int(info[1])] = info[2]
    for i in range(0,upperlimit):
        if i in tmp_dict:
            summary_dict[i].append(tmp_dict[i])
        else:
            summary_dict[i].append('0')

output.write('distance\t' + '\t'.join(name_list) + '\n')

for i in range(0,upperlimit):
    output.write(str(i) + '\t' + '\t'.join(summary_dict[i]) + '\n')
```

do_summary_table.py

```
#!/usr/bin/python
#Author. Ismael Padioleau
#16 february 2012
#take 3 args:
#   A qc_mapping directory, that contain map_stats.csv file for each sample
#   An exon_coverage_summary file (usualy found in exon_quantification directory)
#   An output file to write the summary table

import os
import sys
import fnmatch

input_qc = sys.argv[1]
input_ex = sys.argv[2]
output = open(sys.argv[3],'a')

#check input
if not os.path.exists(input_qc):
    sys.stderr.write('ERROR: The given qc directory does not exist\n')
    sys.exit(1)

if not os.path.isfile(input_ex):
    sys.stderr.write('ERROR: The given exon_coverage_summary file, does not exist\n'
)
    sys.exit(1)

#get info from exon_coverage_summary
ex_quant = open(input_ex,'r')
names = ex_quant.readline().strip().split('\t')
tmp = ex_quant.readline()
tmp = ex_quant.readline()
chro =  ex_quant.readline().strip().split('\t')
exo = ex_quant.readline().strip().split('\t')
ex_quant.close()

#process all files from qc_mapping
qc_list = os.listdir(input_qc)
cpt_file = 0
```

```
output.write('ID,sampleID,date,lane,total.read,mapped.read,read.one.mapped,read.two.mapped,read.unmapped,proper.pair,unique.mapping,multiple.mapping,mapQ=0,mapQ>10,unique.and.mapQ=0,mapQ>10.and.proper.pair,chromosomal.read,exonic.read\n')
for name in qc_list:
    if fnmatch.fnmatch(name, '*map_stats.csv'):
        cpt_file += 1
        tmp = open(input_qc + '/' + name,'r')
        line = tmp.readline()
        line = tmp.readline().strip()
        info = name.split('_')
        ID = '_'.join(info[0:len(info)-3])
        sampleID = '_'.join(info[0:len(info)-5])
        date = info[len(info)-5]
        lane = info[len(info)-4]
        if ID in names:
            index = names.index(ID)+1
            exonic_read = str(exo[index])
            chromosomal_read = str(chro[index])
        else:
            exonic_read = 'NA'
            chromosomal_read = 'NA'
        output.write(ID + ',' + sampleID + ',' + date + ',' + lane + ',' + line + ',' + chromosomal_read + ',' + exonic_read + '\n')

output.close()
if cpt_file == 0:
    sys.stderr.write('ERROR: The given qc directory, contain no stat file to be processed\n')
    sys.exit(1)
```

get_mapping_info.py

```
#! /home/ipadiole/bin python
#Write insert size info in output file
#Take 3 arg:input_sam_file, outputfile, outputfile2

import sys

#Input and output files
output_file = open(sys.argv[1],'a') # Will contain unique mapping, repeat mapping, Q=0 and q>10 count...
output_file2 = ''
if len(sys.argv) > 2:
    output_file2 = open(sys.argv[2],'a') #Will contain insert-size count info

#variables
dict_of_insert_size = {}
q0_count = 0
q_sup_10_count = 0
unique_count = 0
unique_and_q0 = 0
repeat_count = 0
m_count = 0
read_count = 0
proper_pair = 0
q10_proper_pair = 0
read_mapped = 0
read_one_mapped = 0
read_two_mapped = 0
read_unmapped = 0
u = 0

# A function to read bitwise flag
```

```
def readflag(samFlag,proper_pair_flag,proper_pair,read_one_mapped,read_two_mapped,read_unmapped):
    flag = bin(samFlag)[2:]
    flag = "".join(reversed(flag))
    if len(flag)>1:
        if flag[1] == '1':
            proper_pair += 1
            proper_pair_flag = 1
            if len(flag)>5:
                if flag[6] == '1':
                    read_one_mapped += 1
                if flag[6] == '0':
                    read_two_mapped += 1
        elif flag[2] == '1':
            read_unmapped += 1

    return(proper_pair_flag,proper_pair,read_one_mapped,read_two_mapped,read_unmapped)

for line in sys.stdin:
    # Skip headers
    if line[0] == '@':
        continue
    else:
        proper_pair_flag = 0
        # For line in input_file:
        u = 0
        info = line.split('\t')
        two_read_size = len(info[9]) * 2
        read_count += 1
        proper_pair_flag,proper_pair,read_one_mapped,read_two_mapped,read_unmapped = readflag(int(info[1]),proper_pair_flag,proper_pair,read_one_mapped,read_two_mapped,read_unmapped)
        if 'XT:A:U' in info:
            unique_count +=1
            u = 1
        if 'XT:A:R' in info:
            repeat_count +=1
        if not int(info[4]) == 0:
            if int(info[4])> 10:
                q_sup_10_count += 1
                if(proper_pair_flag == 1):
                    q10_proper_pair +=1
        else:
            q0_count += 1
            if u == 1:
                unique_and_q0 += 1

        if not info[0] in dict_of_insert_size and int(info[8]) != 0:
            insert_size = abs(int(info[8])) - two_read_size
            dict_of_insert_size[info[0]] = insert_size

list_of_values = dict_of_insert_size.values()

list_of_values.sort()

if output_file2 != '':
    dict_of_insertsize = {}
    for value in list_of_values:
        if not value in dict_of_insertsize:
            dict_of_insertsize[value] = 1
        else:
            dict_of_insertsize[value] += 1
    output_file2.write('insert size,count\n')
    for item in dict_of_insertsize.items():
        output_file2.write(str(item[0]) + ',' + str(item[1]) + '\n')
```

```
93.
94. output_file.write('Total read,Mapped,Read one mapped, Read two mapped,Read unmapped,
    Proper pair,Unique mapping,Multiple mapping,Q=0,Q>10,unique and Q=0,Q>10 and proper
    pair\n')
95. output_file.write(str(read_count) + ',' + str(read_count -
     read_unmapped) + ',' + str(read_one_mapped) + ',' + str(read_two_mapped) + ',' + st
    r(read_unmapped) + ',' + str(proper_pair) + ',' + str(unique_count) + ',' + str(repe
    at_count) + ',' + str(q0_count) + ',' + str(q_sup_10_count) + ',' + str(unique_and_q
    0) + ',' + str(q10_proper_pair) + '\n')
```

runFastqQuality.py

```
1.  #!/usr/bin/env python
2.  # Author: Ismael Padioleau
3.  # Contact : ismael.padioleau@igh.cnrs.fr
4.  # May 19th 2014
5.  # Runs qc_fastq
6.
7.  #imports
8.  import os
9.  import sys
10.
11. # Global function, import it to run qc_fastq
12. def runFastqQuality_command(basename, configDict):
13.     pairend = configDict.get('pairend')
14.     uid = configDict.get('uid')
15.     task = configDict.get('task')
16.     fastqDir = configDict.get('fastq_dir')
17.     qcFastqDir = configDict.get('qc_fastq_dir')
18.     logDir = qcFastqDir + '/log/'
19.     fastxDir = qcFastqDir + '/fastx/'
20.     read_quality_report = qcFastqDir + '/read_quality_report.csv'
21.     scriptDir = qcFastqDir + '/script/'
22.     project = configDict.get('project')
23.     info = basename.split('_')
24.     # Create log dir if does not exits
25.     if not os.path.isdir(logDir):
26.         os.makedirs(logDir)
27.     if not os.path.exists(scriptDir):
28.        os.makedirs(scriptDir)
29.     if not os.path.exists(fastxDir):
30.        os.makedirs(fastxDir)
31.     if not os.path.exists(read_quality_report):
32.        report = open(read_quality_report, 'w')
33.        report.write("sample total unique %_unique most_represented copy_of_most_repr
    esented %_of_most_represented\n")
34.        report.close()
35.     # Create command
36.     names = [basename + '_1.fastq.gz']
37.     if pairend == '1':
38.         names.append(basename + '_2.fastq.gz')
39.     for name in names:
40.         tmp_name = name.split('.fastq.gz')[0]
41.         job_basename = 'job_' + uid + '_qc_fastq_' + tmp_name
42.         stat_file = fastxDir + tmp_name + '.stat'
43.         tmp_report = qcFastqDir + '/' + tmp_name + '_tmp_report.csv'
44.         # Write and execute the script
45.         script = open(scriptDir + name + '_qc_fastq.sh', 'w')
46.         script.write('#!/bin/bash\n#$ -m a\n#$ -q workq\n')
47.         script.write('#$ -o ' + logDir + tmp_name + '_qc_fastq.out\n')
48.         script.write('#$ -e ' + logDir + tmp_name + '_qc_fastq.err\n')
49.         script.write('#$ -l h_vmem=20G\n#$ -l mem=20G\n')
50.         script.write('#$ -N ' + job_basename + '\n')
51.         script.write('# FASTX commands\n')
```

```
52.         script.write('fastx_quality_stats -Q 33 -i <(gunzip -
    c ' + fastqDir + '/' + name + ') -o ' + stat_file + '\n')
53.         script.write('fastq_quality_boxplot_graph.sh -i ' + stat_file + ' -
    o ' + fastxDir + '/' + tmp_name + '_quality.png -t ' + tmp_name + '\n')
54.         script.write('fastx_nucleotide_distribution_graph.sh -i ' + stat_file + ' -
    o ' + fastxDir + '/' + tmp_name + '_nuc.png -t ' + tmp_name + '\n')
55.         script.write('rm ' + stat_file + '\n')
56.         script.write("echo -en '" + tmp_name + "\\t' >> " + tmp_report + "\n")
57.         script.write("gunzip -c " + fastqDir + '/' + name + " | awk '((NR-
    2)%4 == 0){read=$1;total ++; count[read] ++}END{for(read in count) {if(!max||count[r
    ead]>max) {max=count[read];maxRead=read}; if(count[read] == 1){unique ++}};print tot
    al,unique,unique*100/total,maxRead,count[maxRead],count[maxRead]*100/total}' >> " +
    tmp_report + '\n')
58.         script.write("cat " + tmp_report + " >> " + read_quality_report + '\n')
59.         script.write("rm " + tmp_report + "\n")
60.         script.write('\n#end of the script\n')
61.         script.close()
62.         os.system('qsub ' + scriptDir + name + '_qc_fastq.sh')
63.
64.
65. ###
66. def runFastqQuality(configDict):
67.     print('QC fastq for project : ' + configDict['project'])
68.     for name in configDict['fastq_list']:
69.         runFastqQuality_command(name,configDict)
70.     return configDict
```

runMappingQuality.py

```
1.  #!/usr/bin/env python
2.  # Author: Ismael Padioleau
3.  # May 19th 2014
4.  # Runs qc_mapping for quality
5.  # Functions exported: runMappingQuality
6.
7.  #imports
8.  import os
9.  import sys
10.
11. def runMappingQuality_command(basename,configDict):
12.     print('My base name : ' + basename)
13.     distance_script = configDict.get('distance_between_reads_script')
14.     samtools = configDict.get('samtools')
15.     pairend = configDict.get('pairend')
16.     uid = configDict.get('uid')
17.     task = configDict.get('task')
18.     bamDir = configDict.get('bam_dir')
19.     qcMappingDir = configDict.get('qc_mapping_dir')
20.     logDir = qcMappingDir + '/log/'
21.     scriptDir = qcMappingDir + '/script/'
22.     project = configDict.get('project')
23.     if 'quality_threshold_dist_between_reads' in configDict:
24.         qual_for_distance_between_reads = configDict.get('quality_threshold_dist_bet
    ween_reads')
25.     else:
26.         qual_for_distance_between_reads = '0'
27.     job_basename = 'job_' + uid + '_qc_mapping_' + basename
28.
29.     # Create directories
30.     if not os.path.isdir(logDir):
31.         os.makedirs(logDir)
32.     if not os.path.isdir(scriptDir):
33.         os.makedirs(scriptDir)
34.
```

```
    # Set waiting condition if required
    wait_condition = '\n'
    if ('t1.1_run' in configDict or 't1.2_run' in configDict or 't1.3' in configDict
 or 't1.4' in configDict):
        wait_condition = '#$ -
hold_jid job_' + uid + '_fastqToBam_*_' + basename + '\n'

    # Set names
    rawname = basename.replace('-','_')
    rawname = rawname.replace('.','_')
    rawname = rawname.replace(' ','_')
    bamFile = os.path.join(bamDir, basename + '.bam')
    flagstatFile = os.path.join(qcMappingDir, basename + '.flagstat')
    distanceFile = os.path.join(qcMappingDir, basename + '_distance_between_reads.cs
v')
    logFile = os.path.join(logDir, basename + '_qc_mapping.out')
    errFile = os.path.join(logDir, basename + '_qc_mapping.err')
    scriptFile = os.path.join( scriptDir, basename + '_mapping_qc.sh')
    infoReadCountFile = os.path.join(qcMappingDir, 'info_read_count.csv')
    temporaryInfoReadCountFile = os.path.join(qcMappingDir, basename + '_info_read_c
ount_TMP.csv')
    infoReadUnmappedFile = os.path.join(qcMappingDir, 'info_unmapped_read.csv')
    temporaryInfoReadUnmappedFile = os.path.join(qcMappingDir, basename + 'info_unma
pped_read_TMP.csv')

    # Create count file if necesary
    if not os.path.isfile(infoReadCountFile):
        countFile = open(infoReadCountFile, 'w')
        countFile.write("Sample\tRaw reads\tMapped reads\tPercent mapped reads\tUniq
ue genome coordinates mapped\tPercent of unique mapped position\tMost mapped positio
n\tMaximum reads mapping the same position\tPercent of reads mapped at the most mapp
ed position\n")
        countFile.close()

    # Write script
    script = open(scriptFile, 'w')
    script.write('#!/bin/bash\n#$ -m a\n#$ -q workq\n')
    script.write('#$ -l h_vmem=15G\n#$ -l mem=15G\n')
    script.write('#$ -o ' + logFile + '\n')
    script.write('#$ -e ' + errFile + '\n')
    script.write('#$ -N ' + job_basename + '\n')
    script.write(wait_condition)
    script.write('\n# FLAGSTAT\n')
    script.write(samtools + ' flagstat ' + bamFile + ' > ' + flagstatFile + '\n')
    script.write('\n# READS MAPPED COUNT\n')
    script.write("echo -
en '" + basename + "\\t' >> " + temporaryInfoReadCountFile + '\n')
    script.write("raw_" + rawname + "=$(grep total " + flagstatFile + " | awk '{prin
t $1}')\n")
    script.write("bamToBed -i " + bamFile + " | awk -vRAW=$raw_" + rawname + " -
F $'\\t' ' {coordinates=$1\":\"$2\"-
\"$3;total++;count[coordinates]++}END{for(coordinates in count){if(!max||count[coord
inates]>max){max=count[coordinates];maxCoor=coordinates};if(count[coordinates]==1){u
nique++}};print RAW,\"\\t\",total,\"\\t\",total*100/RAW,\"\\t\",unique,\"\\t\",uniqu
e*100/total,\"\\t\",maxCoor,\"\\t\",count[maxCoor],\"\\t\",count[maxCoor]*100/total}
' >> " + temporaryInfoReadCountFile + "\n")
    script.write("cat " + temporaryInfoReadCountFile + " >> " + infoReadCountFile +
"\n")
    script.write("rm " + temporaryInfoReadCountFile + "\n")
    script.write('\n# UNMAPPED INFO\n')
    script.write("echo -
en " + basename + "'\\t'>> " + temporaryInfoReadUnmappedFile + "\n")
    script.write("samtools view -
f 0x0004 " + bamFile + " | awk '{read=$10;total++;count[read]++}END{print \"Total_no
n-mapped_reads\",total;for(read in count){print read,count[read]+0}}' | sort -
k2,2nr | head -11 >> " + temporaryInfoReadUnmappedFile + "\n")
```

```
    script.write("cat " + temporaryInfoReadUnmappedFile + " >> " + infoReadUnmappedFile + "\n")
    script.write("rm  " + temporaryInfoReadUnmappedFile + "\n")
    script.write('\n# DISTANCE BETWEEN READS\n')
    script.write(samtools + " view " + bamFile + " | python " + distance_script + ' ' + distanceFile + ' ' + qual_for_distance_between_reads + '\n')
    script.write('\n#end of the script\n')
    script.close()
    os.system('qsub ' + scriptFile)


###
def runMappingQuality(configDict):
    print('QC mapping for project : ' + configDict['project'])
    for name in configDict['bam_list']:
        runMappingQuality_command(name,configDict)
    # Run summary report
    distance_script = configDict.get('distance_between_reads_summary')
    scriptFile = configDict.get('qc_mapping_dir') + '/script/summary_qc_report.sh'
    logFile = configDict.get('qc_mapping_dir') + '/log/summary_qc_report.out'
    errFile = configDict.get('qc_mapping_dir') + '/log/summary_qc_report.err'
    if not 'dist_summury_upper_limit' in configDict:
                upper_limit = '150'
            else:
                upper_limit = configDict('dist_summury_upper_limit')
            script = open(scriptFile, 'w')
            script.write('#!/bin/bash\n#$ -m a\n#$ -q workq\n')
            script.write('#$ -o ' + logFile + '\n')
            script.write('#$ -e ' + errFile + '\n')
            script.write('#$ -hold_jid job_' + configDict.get('uid') + '_qc_mapping_*' + '\n')
            script.write('#$ -N job_' + configDict.get('uid') + '_report_qc\n')
            if configDict.get('pairend') == '1':
                script.write('find ' + configDict.get('qc_mapping_dir') + ' -name *.flagstat -exec grep -H properly "{}" \; > ' + configDict.get('qc_mapping_dir') + '/flagstat_report.txt\n')
            else:
                script.write('find ' + configDict.get('qc_mapping_dir') + ' -name *.flagstat -exec grep -H \'mapped (\' "{}" \; > ' + configDict.get('qc_mapping_dir') + '/flagstat_report.txt\n')
            script.write('python ' + distance_script + ' ' + configDict.get('qc_mapping_dir') + ' ' + configDict.get('qc_mapping_dir') + '/distance_between_reads_summary.csv + ' + upper_limit + '\n')
            script.write('\n#end of the script\n')
            script.close()
            os.system('qsub ' + scriptFile)
            return configDict

        def runSummaryTable(configDict):
            #do summary table for the run
            uid = configDict['uid']
            script_path = configDict['qc_summary_script']
            infoDir = configDict['info_dir']
            qcMappingDir = configDict.get('qc_mapping_dir')
            logDir = infoDir + '/log'
            exon_quant_summary = configDict['eq_dir'] + '/exon_coverage_summary'
            qcMappingDir = configDict.get('qc_mapping_dir')
            task_list = configDict['task']
            if not os.path.isdir(logDir):
                os.makedirs(logDir)
            wait_condition = ' '
            if (('3' in task_list) and ('5' in task_list or '5.1' in task_list )):
                wait_condition = '-w \'ended(' + configDict['uid'] + '*_quantification_*' + configDict['project'] + '*)
```

```
    && ended(' + configDict['uid'] + '*_qc_mapping_*' + configDict['project'] + '*)\' '

            elif ('3' in task_list):
                wait_condition = '-w \'ended(' + configDict['uid'] + '*_quantification_*' + configDict['project'] + '*)\' '
            elif('5' in task_list or '5.1' in task_list ):
                wait_condition = '-w \'ended(' + configDict['uid'] + '*_qc_mapping_*' + configDict['project'] + '*)\' '

            command = 'python ' + script_path + ' ' + qcMappingDir + ' ' + exon_quant_summary + ' ' + infoDir + '/summary_table'
            lsf_command = 'bsub -q normal '+ wait_condition + ' -J ' + uid + '_qc_summary -o ' + os.path.join(logDir,'summary.out') + ' ' + command
            os.system(lsf_command)
            return configDict
```

BIBLIOGRAPHIE

Abdurashidova, G., Radulescu, S., Sandoval, O., Zahariev, S., Danailov, M.B., Demidovich, A., Santamaria, L., Biamonti, G., Riva, S., and Falaschi, A. (2007). Functional interactions of DNA topoisomerases with a human replication origin. The EMBO Journal *26*, 998–1009.

Aguilera, A. (2002). The connection between transcription and genomic instability. EMBO J *21*, 195–201.

Aguilera, A., and García-Muse, T. (2012). R Loops: From Transcription Byproducts to Threats to Genome Stability. Molecular Cell *46*, 115–124.

Aladjem, M.I. (2007). Replication in context: dynamic regulation of DNA replication patterns in metazoans. Nat Rev Genet *8*, 588–600.

Alberini, C.M. (2009). Transcription Factors in Long-Term Memory and Synaptic Plasticity. Physiol Rev *89*.

Alén, C., Kent, N.A., Jones, H.S., O'Sullivan, J., Aranda, A., and Proudfoot, N.J. (2002). A Role for Chromatin Remodeling in Transcriptional Termination by RNA Polymerase II. Molecular Cell *10*, 1441–1452.

Ammazzalorso, F., Pirzio, L.M., Bignami, M., Franchitto, A., and Pichierri, P. (2010). ATR and ATM differently regulate WRN to prevent DSBs at stalled replication forks and promote replication fork recovery. The EMBO Journal *29*, 3156–3169.

Anczuków, O., Akerman, M., Cléry, A., Wu, J., Shen, C., Shirole, N.H., Raimer, A., Sun, S., Jensen, M.A., Hua, Y., et al. (2015). SRSF1-Regulated Alternative Splicing in Breast Cancer. Molecular Cell *60*, 105–117.

Anders, S., Pyl, P.T., and Huber, W. (2015). HTSeq—a Python framework to work with high-throughput sequencing data. Bioinformatics *31*, 166–169.

Andersson, R., Andersen, P.R., Valen, E., Core, L.J., Bornholdt, J., Boyd, M., Jensen, T.H., and Sandelin, A. (2014). Nuclear stability and transcriptional directionality separate functionally distinct RNA species. Nature Communications *5*, ncomms6336.

Azvolinsky, A., Giresi, P.G., Lieb, J.D., and Zakian, V.A. (2009). Highly Transcribed RNA Polymerase II Genes Are Impediments to Replication Fork Progression in Saccharomyces cerevisiae. Molecular Cell *34*, 722–734.

Bailey, T.L., and Machanick, P. (2012). Inferring direct DNA binding from ChIP-seq. Nucleic Acids Res *40*, e128–e128.

Bailey, T., Krajewski, P., Ladunga, I., Lefebvre, C., Li, Q., Liu, T., Madrigal, P., Taslim, C., and Zhang, J. (2013). Practical Guidelines for the Comprehensive Analysis of ChIP-seq Data. PLOS Computational Biology *9*, e1003326.

Baranello, L., Wojtowicz, D., Cui, K., Devaiah, B.N., Chung, H.-J., Chan-Salis, K.Y., Guha, R., Wilson, K., Zhang, X., Zhang, H., et al. (2016). RNA Polymerase II Regulates Topoisomerase 1 Activity to Favor Efficient Transcription. Cell *165*, 357–371.

Barlow, J.H., and Nussenzweig, A. (2014). Replication initiation and genome instability: a crossroads for DNA and RNA synthesis. Cell. Mol. Life Sci. *71*, 4545–4559.

Barlow, J., Faryabi, R.B., Callen, E., Wong, N., Malhowski, A., Chen, H.T., Gutierez-Cruz, G., Sun, H.-W., McKinnon, P., Wright, G., et al. (2013). A novel class of early replicating fragile sites that contribute to genome instability in B cell lymphomas. Cell *152*, 620–632.

Barski, A., Cuddapah, S., Cui, K., Roh, T.-Y., Schones, D.E., Wang, Z., Wei, G., Chepelev, I., and Zhao, K. (2007). High-resolution profiling of histone methylations in the human genome. Cell *129*, 823–837.

Bell, S.D. (2006). Molecular biology: Prime-time progress. Nature *439*, 542–543.

Bell, S.P. (2002). The origin recognition complex: from simple origins to complex functions. Genes Dev. *16*, 659–672.

Bell, S.P., and Dutta, A. (2002). DNA Replication in Eukaryotic Cells. Annual Review of Biochemistry *71*, 333–374.

Bermejo, R., Doksani, Y., Capra, T., Katou, Y.-M., Tanaka, H., Shirahige, K., and Foiani, M. (2007). Top1- and Top2-mediated topological transitions at replication forks ensure fork progression and stability and prevent DNA damage checkpoint activation. Genes Dev. *21*, 1921–1936.

Bermejo, R., Lai, M.S., and Foiani, M. (2012). Preventing Replication Stress to Maintain Genome Stability: Resolving Conflicts between Replication and Transcription. Molecular Cell *45*, 710–718.

Bezine, E., Vignard, J., and Mirey, G. (2014). The Cytolethal Distending Toxin Effects on Mammalian Cells: A DNA Damage Perspective. Cells *3*, 592–615.

Bhatia, V., Barroso, S.I., García-Rubio, M.L., Tumini, E., Herrera-Moyano, E., and Aguilera, A. (2014). BRCA2 prevents R-loop accumulation and associates with TREX-2 mRNA export factor PCID2. Nature *511*, 362–365.

Blackford, A.N., and Jackson, S.P. (2017). ATM, ATR, and DNA-PK: The Trinity at the Heart of the DNA Damage Response. Molecular Cell *66*, 801–817.

Blumenthal, A.B., Kriegstein, H.J., and Hogness, D.S. (1974). The units of DNA replication in Drosophila melanogaster chromosomes. Cold Spring Harb. Symp. Quant. Biol. 38, 205–223.

Boehm, A.K., Saunders, A., Werner, J., and Lis, J.T. (2003). Transcription Factor and Polymerase Recruitment, Modification, and Movement on dhsp70 In Vivo in the Minutes following Heat Shock. Mol Cell Biol *23*, 7628–7637.

Boguslawski, S.J., Smith, D.E., Michalak, M.A., Mickelson, K.E., Yehle, C.O., Patterson, W.L., and Carrico, R.J. (1986). Characterization of monoclonal antibody to DNA · RNA and its application to immunodetection of hybrids. Journal of Immunological Methods *89*, 123–130.

Boque-Sastre, R., Soler, M., Oliveira-Mateos, C., Portela, A., Moutinho, C., Sayols, S., Villanueva, A., Esteller, M., and Guil, S. (2015). Head-to-head antisense transcription and R-loop formation promotes transcriptional activation. Proc Natl Acad Sci U S A *112*, 5785–5790.

Brill, S.J., DiNardo, S., Voelkel-Meiman, K., and Sternglanz, R. (1987). Need for DNA topoisomerase activity as a swivel for DNA replication for transcription of ribosomal RNA. Nature *326*, 414–416.

Buratowski, S., Hahn, S., Sharp, P.A., and Guarente, L. (1988). Function of a yeast TATA element-binding protein in a mammalian transcription system. Nature *334*, 37–42.

Cáceres, J.F., Screaton, G.R., and Krainer, A.R. (1998). A specific subset of SR proteins shuttles continuously between the nucleus and the cytoplasm. Genes Dev. *12*, 55–66.

Cadoret, J.-C., Meisch, F., Hassan-Zadeh, V., Luyten, I., Guillet, C., Duret, L., Quesneville, H., and Prioleau, M.-N. (2008). Genome-wide studies highlight indirect links between human replication origins and gene regulation. PNAS *105*, 15837–15842.

Castellano-Pozo, M., Santos-Pereira, J.M., Rondón, A.G., Barroso, S., Andújar, E., Pérez-Alegre, M., García-Muse, T., and Aguilera, A. (2013). R Loops Are Linked to Histone H3 S10 Phosphorylation and Chromatin Condensation. Molecular Cell *52*, 583–590.

Cerritelli, S.M., and Crouch, R.J. (2009). Ribonuclease H: the enzymes in Eukaryotes. FEBS J *276*, 1494–1505.

Chakraborty, P., and Grosse, F. (2011). Human DHX9 helicase preferentially unwinds RNA-containing displacement loops (R-loops) and G-quadruplexes. DNA Repair *10*, 654–665.

Champoux, J.J. (2001). DNA Topoisomerases: Structure, Function, and Mechanism. Annual Review of Biochemistry *70*, 369–413.

Chen, S.H., Chan, N.-L., and Hsieh, T. (2013). New Mechanistic and Functional Insights into DNA Topoisomerases. Annual Review of Biochemistry *82*, 139–170.

Clayton, R.A., White, O., Ketchum, K.A., and Venter, J.C. (1997). The first genome from the third domain of life. Nature *387*, 459–462.

Conaway, J.W., Shilatifard, A., Dvir, A., and Conaway, R.C. (2000). Control of elongation by RNA polymerase II. Trends in Biochemical Sciences *25*, 375–380.

Conesa, A., Madrigal, P., Tarazona, S., Gomez-Cabrero, D., Cervera, A., McPherson, A., Szcześniak, M.W., Gaffney, D.J., Elo, L.L., Zhang, X., et al. (2016). A survey of best practices for RNA-seq data analysis. Genome Biol *17*.

Core, L.J., Waterfall, J.J., and Lis, J.T. (2008). Nascent RNA sequencing reveals widespread pausing and divergent initiation at human promoters. Science *322*, 1845–1848.

Cortez, D., Guntuku, S., Qin, J., and Elledge, S.J. (2001). ATR and ATRIP: Partners in Checkpoint Signaling. Science *294*, 1713–1716.

Cory, S., Marcker, K.A., Dube, S.K., and Clark, B.F. (1968). Primary structure of a methionine transfer RNA from Escherichia coli. Nature *220*, 1039–1040.

Coster, G., Frigola, J., Beuron, F., Morris, E.P., and Diffley, J.F.X. (2014). Origin Licensing Requires ATP Binding and Hydrolysis by the MCM Replicative Helicase. Molecular Cell *55*, 666–677.

Couch, F.B., Bansbach, C.E., Driscoll, R., Luzwick, J.W., Glick, G.G., Bétous, R., Carroll, C.M., Jung, S.Y., Qin, J., Cimprich, K.A., et al. (2013). ATR phosphorylates SMARCAL1 to prevent replication fork collapse. Genes Dev. *27*, 1610–1623.

Cramer, P. (2004). RNA polymerase II structure: from core to functional complexes. Current Opinion in Genetics & Development *14*, 218–226.

Crosetto, N., Mitra, A., Silva, M.J., Bienko, M., Dojer, N., Wang, Q., Karaca, E., Chiarle, R., Skrzypczak, M., Ginalski, K., et al. (2013). Nucleotide-resolution DNA double-strand break mapping by next-generation sequencing. Nat Meth *10*, 361–365.

Dahmus, M.E. (1996). Reversible Phosphorylation of the C-terminal Domain of RNA Polymerase II. J. Biol. Chem. *271*, 19009–19012.

Danis, E., Brodolin, K., Menut, S., Maiorano, D., Girard-Reydet, C., and Méchali, M. (2004). Specification of a DNA replication origin by a transcription complex. Nat Cell Biol *6*, 721–730.

De, I., Bessonov, S., Hofele, R., dos Santos, K., Will, C.L., Urlaub, H., Lührmann, R., and Pena, V. (2015). The RNA helicase Aquarius exhibits structural adaptations mediating its recruitment to spliceosomes. Nat Struct Mol Biol *22*, 138–144.

Diffley, J.F.X. (2004). Regulation of Early Events in Chromosome Replication. Current Biology *14*, R778–R786.

Dobin, A., Davis, C.A., Schlesinger, F., Drenkow, J., Zaleski, C., Jha, S., Batut, P., Chaisson, M., and Gingeras, T.R. (2013). STAR: ultrafast universal RNA-seq aligner. Bioinformatics *29*, 15–21.

Drolet, M., Phoenix, P., Menzel, R., Massé, E., Liu, L.F., and Crouch, R.J. (1995). Overexpression of RNase H partially complements the growth defect of an Escherichia coli delta topA mutant: R-loop formation is a major problem in the absence of DNA topoisomerase I. Proc Natl Acad Sci U S A *92*, 3526–3530.

Dube, S.K., Marcker, K.A., Clark, B.F., and Cory, S. (1968). Nucleotide sequence of N-formyl-methionyl-transfer RNA. Nature *218*, 232–233.

Dungrawala, H., Rose, K.L., Bhat, K.P., Mohni, K.N., Glick, G.G., Couch, F.B., and Cortez, D. (2015). The Replication Checkpoint Prevents Two Types of Fork Collapse without Regulating Replisome Stability. Molecular Cell *59*, 998–1010.

Duquette, M.L., Handa, P., Vincent, J.A., Taylor, A.F., and Maizels, N. (2004). Intracellular transcription of G-rich DNAs induces formation of G-loops, novel structures containing G4 DNA. Genes Dev. *18*, 1618–1629.

Dutrow, N., Nix, D.A., Holt, D., Milash, B., Dalley, B., Westbroek, E., Parnell, T.J., and Cairns, B.R. (2008). Dynamic transcriptome of Schizosaccharomyces pombe shown by RNA-DNA hybrid mapping. Nat Genet *40*, 977–986.

Eder, P.S., and Walder, J.A. (1991). Ribonuclease H from K562 human erythroleukemia cells. Purification, characterization, and substrate specificity. J. Biol. Chem. *266*, 6472–6479.

Engström, P.G., Steijger, T., Sipos, B., Grant, G.R., Kahles, A., Rätsch, G., Goldman, N., Hubbard, T.J., Harrow, J., Guigó, R., et al. (2013). Systematic evaluation of spliced alignment programs for RNA-seq data. Nat Methods *10*, 1185–1191.

Falaschi, A. (2009). Binding of DNA Topoisomerases I and II to Replication Origins. In DNA Topoisomerases, (Humana Press, Totowa, NJ), pp. 131–143.

Fiers, W., Contreras, R., Duerinck, F., Haegeman, G., Iserentant, D., Merregaert, J., Min Jou, W., Molemans, F., Raeymaekers, A., Van den Berghe, A., et al. (1976). Complete nucleotide sequence of bacteriophage MS2 RNA: primary and secondary structure of the replicase gene. Nature *260*, 500–507.

Gan, W., Guan, Z., Liu, J., Gui, T., Shen, K., Manley, J.L., and Li, X. (2011). R-loop-mediated genomic instability is caused by impairment of replication fork progression. Genes Dev. *25*, 2041–2056.

García-Muse, T., and Aguilera, A. (2016). Transcription-replication conflicts: how they occur and how they are resolved. Nat Rev Mol Cell Biol *17*, 553–563.

García-Rubio, M.L., Pérez-Calero, C., Barroso, S.I., Tumini, E., Herrera-Moyano, E., Rosado, I.V., and Aguilera, A. (2015). The Fanconi Anemia Pathway Protects Genome Integrity from R-loops. PLOS Genetics *11*, e1005674.

Gilbert, D.M. (2002). Replication timing and transcriptional control: beyond cause and effect. Current Opinion in Cell Biology *14*, 377–383.

Ginno, P.A., Lott, P.L., Christensen, H.C., Korf, I., and Chédin, F. (2012). R-Loop Formation Is a Distinctive Characteristic of Unmethylated Human CpG Island Promoters. Molecular Cell *45*, 814–825.

Ginno, P.A., Lim, Y.W., Lott, P.L., Korf, I., and Chédin, F. (2013). GC skew at the 5′ and 3′ ends of human genes links R-loop formation to epigenetic regulation and transcription termination. Genome Res *23*, 1590–1600.

Haas, B.J., and Zody, M.C. (2010). Advancing RNA-Seq analysis. Nat Biotech *28*, 421–423.

Halazonetis, T.D., Gorgoulis, V.G., and Bartek, J. (2008). An Oncogene-Induced DNA Damage Model for Cancer Development. Science *319*, 1352–1355.

Hamperl, S., and Cimprich, K.A. (2014). The contribution of co-transcriptional RNA:DNA hybrid structures to DNA damage and genome instability. DNA Repair (Amst) *19*, 84–94.

Hamperl, S., and Cimprich, K.A. (2016). Conflict Resolution in the Genome: How Transcription and Replication Make It Work. Cell *167*, 1455–1467.

Hamperl, S., Bocek, M.J., Saldivar, J.C., Swigut, T., and Cimprich, K.A. (2017). Transcription-Replication Conflict Orientation Modulates R-Loop Levels and Activates Distinct DNA Damage Responses. Cell *170*, 774–786.e19.

Hansen, R.S., Thomas, S., Sandstrom, R., Canfield, T.K., Thurman, R.E., Weaver, M., Dorschner, M.O., Gartler, S.M., and Stamatoyannopoulos, J.A. (2010). Sequencing newly replicated DNA reveals widespread plasticity in human replication timing. PNAS *107*, 139–144.

Hardcastle, T.J., and Kelly, K.A. (2010). baySeq: Empirical Bayesian methods for identifying differential expression in sequence count data. BMC Bioinformatics *11*, 422.

Hashimoto, Y., Puddu, F., and Costanzo, V. (2012). RAD51- and MRE11-dependent reassembly of uncoupled CMG helicase complex at collapsed replication forks. Nat Struct Mol Biol *19*, 17–24.

Hazra, B., Das Sarma, M., and Sanyal, U. (2004). Separation methods of quinonoid constituents of plants used in Oriental traditional medicines. Journal of Chromatography B *812*, 259–275.

Heidemann, M., Hintermair, C., Voß, K., and Eick, D. (2013). Dynamic phosphorylation patterns of RNA polymerase II CTD during transcription. Biochimica et Biophysica Acta (BBA) - Gene Regulatory Mechanisms *1829*, 55–62.

Heller, R.C., and Marians, K.J. (2006). Replication fork reactivation downstream of a blocked nascent leading strand. Nature *439*, 557–562.

Helmrich, A., Ballarino, M., and Tora, L. (2011). Collisions between Replication and Transcription Complexes Cause Common Fragile Site Instability at the Longest Human Genes. Molecular Cell *44*, 966–977.

Helmrich, A., Ballarino, M., Nudler, E., and Tora, L. (2013). Transcription-replication encounters, consequences and genomic instability. Nat Struct Mol Biol *20*, 412–418.

Hiratani, I., Takebayashi, S., Lu, J., and Gilbert, D.M. (2009). Replication timing and transcriptional control: beyond cause and effect—part II. Current Opinion in Genetics & Development *19*, 142–149.

Hodroj, D., Serhal, K., and Maiorano, D. (2017). Ddx19 links mRNA nuclear export with progression of transcription and replication and suppresses genomic instability upon DNA damage in proliferating cells. Nucleus *0*, 1–7.

Holley, R.W., Apgar, J., Merrill, S.H., and Zubkoff, P.L. (1961). NUCLEOTIDE AND OLIGONUCLEOTIDE COMPOSITIONS OF THE ALANINE-, VALINE-, AND TYROSINE-ACCEPTOR "SOLUBLE" RIBONUCLEIC ACIDS OF YEAST. J. Am. Chem. Soc. *83*, 4861–4862.

Holley, R.W., Apgar, J., Everett, G.A., Madison, J.T., Marquisee, M., Merrill, S.H., Penswick, J.R., and Zamir, A. (1965). Structure of a Ribonucleic Acid. Science *147*, 1462–1465.

Holmes, W.F., Braastad, C.D., Mitra, P., Hampe, C., Doenecke, D., Albig, W., Stein, J.L., Wijnen, A.J. van, and Stein, G.S. (2005). Coordinate Control and Selective Expression of the Full Complement of Replication-dependent Histone H4 Genes in Normal and Cancer Cells. J. Biol. Chem. *280*, 37400–37407.

Horak, C.E., and Snyder, M. (2002). ChIP-chip: a genomic approach for identifying transcription factor binding sites. Meth. Enzymol. *350*, 469–483.

Hu, B., Petela, N., Kurze, A., Chan, K.-L., Chapard, C., and Nasmyth, K. (2015). Biological chromodynamics: a general method for measuring protein occupancy across the genome by calibrating ChIP-seq. Nucleic Acids Res *43*, e132–e132.

Hu, H., Baack, M., and Knippers, R. (2009). Proteins of the origin recognition complex (ORC) and DNA topoisomerases on mammalian chromatin. BMC Molecular Biology *10*, 36.

Huang, Y., Gattoni, R., Stévenin, J., and Steitz, J.A. (2003). SR Splicing Factors Serve as Adapter Proteins for TAP-Dependent mRNA Export. Molecular Cell *11*, 837–843.

Huberman, J.A., and Riggs, A.D. (1968). On the mechanism of DNA replication in mammalian chromosomes. J. Mol. Biol. 32, 327–341.

Huertas, P., and Aguilera, A. (2003). Cotranscriptionally Formed DNA:RNA Hybrids Mediate Transcription Elongation Impairment and Transcription-Associated Recombination. Molecular Cell *12*, 711–721.

Huvet, M., Nicolay, S., Touchon, M., Audit, B., d'Aubenton-Carafa, Y., Arneodo, A., and Thermes, C. (2007). Human gene organization driven by the coordination of replication and transcription. Genome Res. *17*, 1278–1285.

Iacovoni, J.S., Caron, P., Lassadi, I., Nicolas, E., Massip, L., Trouche, D., and Legube, G. (2010). High-resolution profiling of γH2AX around DNA double strand breaks in the mammalian genome. EMBO J *29*, 1446–1457.

Ivessa, A.S., Zhou, J.-Q., and Zakian, V.A. (2000). The Saccharomyces Pif1p DNA Helicase and the Highly Related Rrm3p Have Opposite Effects on Replication Fork Progression in Ribosomal DNA. Cell *100*, 479–489.

Jackson, D.A. (1995). S-Phase Progression in Synchronized Human Cells. Experimental Cell Research *220*, 62–70.

Jackson, D.A., and Pombo, A. (1998). Replicon clusters are stable units of chromosome structure: evidence that nuclear organization contributes to the efficient activation and propagation of S phase in human cells. J. Cell Biol. *140*, 1285–1295.

Jaxel, C., Capranico, G., Kerrigan, D., Kohn, K.W., and Pommier, Y. (1991). Effect of local DNA sequence on topoisomerase I cleavage in the presence or absence of camptothecin. J. Biol. Chem. *266*, 20418–20423.

Ji, H., Jiang, H., Ma, W., Johnson, D.S., Myers, R.M., and Wong, W.H. (2008). An integrated software system for analyzing ChIP-chip and ChIP-seq data. Nat Biotech *26*, 1293–1300.

Johnson, D.S., Mortazavi, A., Myers, R.M., and Wold, B. (2007). Genome-Wide Mapping of in Vivo Protein-DNA Interactions. Science *316*, 1497–1502.

Jones, R.M., Mortusewicz, O., Afzal, I., Lorvellec, M., García, P., Helleday, T., and Petermann, E. (2013). Increased replication initiation and conflicts with transcription underlie Cyclin E-induced replication stress. Oncogene *32*, 3744–3753.

Kamada, K., Roeder, R.G., and Burley, S.K. (2003). Molecular mechanism of recruitment of TFIIF- associating RNA polymerase C-terminal domain phosphatase (FCP1) by transcription factor IIF. PNAS *100*, 2296–2299.

Karni, R., de Stanchina, E., Lowe, S.W., Sinha, R., Mu, D., and Krainer, A.R. (2007). The gene encoding the splicing factor SF2/ASF is a proto-oncogene. Nat Struct Mol Biol *14*, 185–193.

Kataoka, N., Bachorik, J.L., and Dreyfuss, G. (1999). Transportin-SR, a Nuclear Import Receptor for SR Proteins. The Journal of Cell Biology *145*, 1145–1152.

Kharchenko, P.V., Tolstorukov, M.Y., and Park, P.J. (2008). Design and analysis of ChIP-seq experiments for DNA-binding proteins. Nat Biotech *26*, 1351–1359.

Kim, J.B., and Sharp, P.A. (2001). Positive Transcription Elongation Factor b Phosphorylates hSPT5 and RNA Polymerase II Carboxyl-terminal Domain Independently of Cyclin-dependent Kinase-activating Kinase. J. Biol. Chem. *276*, 12317–12323.

Kim, R.A., and Wang, J.C. (1989). Function of DNA topoisomerases as replication swivels in Saccharomyces cerevisiae. Journal of Molecular Biology *208*, 257–267.

Kim, H., Kim, J., Selby, H., Gao, D., Tong, T., Phang, T.L., and Tan, A.C. (2011). A short survey of computational analysis methods in analysing ChIP-seq data. Human Genomics *5*, 117.

Kim, Y.-J., Björklund, S., Li, Y., Sayre, M.H., and Kornberg, R.D. (1994). A multiprotein mediator of transcriptional activation and its interaction with the C-terminal repeat domain of RNA polymerase II. Cell *77*, 599–608.

Knott, S.R.V., Viggiani, C.J., and Aparicio, O.M. (2009). To promote and protect: Coordinating DNA replication and transcription for genome stability. Epigenetics *4*, 362–365.

Kohtz, J.D., Jamison, S.F., Will, C.L., Zuo, P., Lührmann, R., Garcia-Blanco, M.A., and Manley, J.L. (1994). Protein–protein interactions and 5'-splice-site recognition in mammalian mRNA precursors. Nature *368*, 119–124.

Koren, A., Polak, P., Nemesh, J., Michaelson, J.J., Sebat, J., Sunyaev, S.R., and McCarroll, S.A. (2012). Differential Relationship of DNA Replication Timing to Different Forms of Human Mutation and Variation. Am J Hum Genet *91*, 1033–1040.

Köster, J., and Rahmann, S. (2012). Snakemake—a scalable bioinformatics workflow engine. Bioinformatics *28*, 2520–2522.

Koundrioukoff, S., Carignon, S., Técher, H., Letessier, A., Brison, O., and Debatisse, M. (2013). Stepwise Activation of the ATR Signaling Pathway upon Increasing Replication Stress Impacts Fragile Site Integrity. PLOS Genetics *9*, e1003643.

Kumar, S., and Huberman, J.A. (2009). Checkpoint-Dependent Regulation of Origin Firing and Replication Fork Movement in Response to DNA Damage in Fission Yeast. Mol Cell Biol *29*, 602–611.

Kunst, F., Ogasawara, N., Moszer, I., Albertini, A.M., Alloni, G., Azevedo, V., Bertero, M.G., Bessières, P., Bolotin, A., Borchert, S., et al. (1997). The complete genome sequence of the Gram-positive bacterium Bacillus subtilis. Nature *390*, 249–256.

Kvam, V.M., Liu, P., and Si, Y. (2012). A comparison of statistical methods for detecting differentially expressed genes from RNA-seq data. Am. J. Bot. *99*, 248–256.

Laajala, T.D., Raghav, S., Tuomela, S., Lahesmaa, R., Aittokallio, T., and Elo, L.L. (2009). A practical comparison of methods for detecting transcription factor binding sites in ChIP-seq experiments. BMC Genomics *10*, 618.

Lander, E.S. (2004). Finishing the euchromatic sequence of the human genome. Nature *431*, 931–945.

Lang, K.S., Hall, A.N., Merrikh, C.N., Ragheb, M., Tabakh, H., Pollock, A.J., Woodward, J.J., Dreifus, J.E., and Merrikh, H. (2017). Replication-Transcription Conflicts Generate R-Loops that Orchestrate Bacterial Stress Survival and Pathogenesis. Cell *170*, 787–799.e18.

Langmead, B., and Salzberg, S.L. (2012). Fast gapped-read alignment with Bowtie 2. Nat Meth *9*, 357–359.

Le Beau, M.M., Rassool, F.V., Neilly, M.E., Espinosa, R., Glover, T.W., Smith, D.I., and McKeithan, T.W. (1998). Replication of a common fragile site, FRA3B, occurs late in S phase and is delayed further upon induction: implications for the mechanism of fragile site induction. Hum. Mol. Genet. *7*, 755–761.

Lefstin, J.A., and Yamamoto, K.R. (1998). Allosteric effects of DNA on transcriptional regulators. Nature *392*, 885–888.

Lemaire, R., Prasad, J., Kashima, T., Gustafson, J., Manley, J.L., and Lafyatis, R. (2002). Stability of a PKCI-1-related mRNA is controlled by the splicing factor ASF/SF2: a novel function for SR proteins. Genes Dev. *16*, 594–607.

Leu, S., Lin, Y.-M., Wu, C.-H., and Ouyang, P. (2012). Loss of Pnn expression results in mouse early embryonic lethality and cellular apoptosis through SRSF1-mediated alternative expression of Bcl-xS and ICAD. J Cell Sci *125*, 3164–3172.

Leuther, K.K., Bushnell, D.A., and Kornberg, R.D. (1996). Two-Dimensional Crystallography of TFIIB– and IIE–RNA Polymerase II Complexes: Implications for Start Site Selection and Initiation Complex Formation. Cell *85*, 773–779.

Li, B., and Dewey, C.N. (2011). RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome. BMC Bioinformatics *12*, 323.

Li, X., and Manley, J.L. (2005). Inactivation of the SR Protein Splicing Factor ASF/SF2 Results in Genomic Instability. Cell *122*, 365–378.

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., and Durbin, R. (2009). The Sequence Alignment/Map format and SAMtools. Bioinformatics *25*, 2078–2079.

Liao, Y., Smyth, G.K., and Shi, W. (2014). featureCounts: an efficient general purpose program for assigning sequence reads to genomic features. Bioinformatics *30*, 923–930.

Lifton, R.P., Goldberg, M.L., Karp, R.W., and Hogness, D.S. (1978). The organization of the histone genes in Drosophila melanogaster: functional and evolutionary implications. Cold Spring Harb. Symp. Quant. Biol. *42 Pt 2*, 1047–1051.

Lin, P.S., Marshall, N.F., and Dahmus, M.E. (2002). CTD phosphatase: role in RNA polymerase II cycling and the regulation of transcript elongation. Prog. Nucleic Acid Res. Mol. Biol. *72*, 333–365.

Liu, L.F., and Wang, J.C. (1987). Supercoiling of the DNA template during transcription. Proc Natl Acad Sci U S A *84*, 7024–7027.

Ljungman, M., and Hanawalt, P.C. (1996). The anti-cancer drug camptothecin inhibits elongation but stimulates initiation of RNA polymerase II transcription. Carcinogenesis *17*, 31–35.

Lossaint, G., Larroque, M., Ribeyre, C., Bec, N., Larroque, C., Décaillet, C., Gari, K., and Constantinou, A. (2013). FANCD2 Binds MCM Proteins and Controls Replisome Function upon Activation of S Phase Checkpoint Signaling. Molecular Cell *51*, 678–690.

Love, M.I., Huber, W., and Anders, S. (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biology *15*, 550.

Luciani, M.G., Oehlmann, M., and Blow, J.J. (2004). Characterization of a novel ATR-dependent, Chk1-independent, intra-S-phase checkpoint that suppresses initiation of replication in Xenopus. Journal of Cell Science *117*, 6019–6030.

Madakashira, B.P., and Sadler, K.C. (2017). DNA Methylation, Nuclear Organization, and Cancer. Front. Genet. *8*.

Maiorano, D., Rul, W., and Méchali, M. (2004). Cell cycle regulation of the licensing activity of Cdt1 in Xenopus laevis. Experimental Cell Research *295*, 138–149.

Malone, B.M., Tan, F., Bridges, S.M., and Peng, Z. (2011). Comparison of Four ChIP-Seq Analytical Algorithms Using Rice Endosperm H3K27 Trimethylation Profiling Data. PLOS ONE *6*, e25260.

Mardis, E.R. (2013). Next-Generation Sequencing Platforms. Annual Review of Analytical Chemistry *6*, 287–303.

Martin, M. (2011). Cutadapt removes adapter sequences from high-throughput sequencing reads. EMBnet.Journal *17*, 10–12.

Massiello, A., and Chalfant, C.E. (2006). SRp30a (ASF/SF2) regulates the alternative splicing of caspase-9 pre-mRNA and is required for ceramide-responsiveness. J. Lipid Res. *47*, 892–897.

McCracken, S., Fong, N., Yankulov, K., Ballantyne, S., Pan, G., Greenblatt, J., Patterson, S.D., Wickens, M., and Bentley, D.L. (1997). The C-terminal domain of RNA polymerase II couples mRNA processing to transcription. Nature *385*, 357–361.

Mellon, I., and Hanawalt, P.C. (1989). Induction of the Escherichia coli lactose operon selectively increases repair of its transcribed DNA strand. Nature *342*, 95–98.

Meryet-Figuiere, M., Alaei-Mahabadi, B., Ali, M.M., Mitra, S., Subhash, S., Pandey, G.K., Larsson, E., and Kanduri, C. (2014). Temporal separation of replication and transcription during S-phase progression. Cell Cycle *13*, 3241–3248.

Miao, Z.-H., Player, A., Shankavaram, U., Wang, Y.-H., Zimonjic, D.B., Lorenzi, P.L., Liao, Z.-Y., Liu, H., Shimura, T., Zhang, H.-L., et al. (2007). Nonclassic Functions of Human Topoisomerase I: Genome-Wide and Pharmacologic Analyses. Cancer Res *67*, 8752–8761.

Min Jou, W., Haegeman, G., Ysebaert, M., and Fiers, W. (1972). Nucleotide sequence of the gene coding for the bacteriophage MS2 coat protein. Nature *237*, 82–88.

Misteli, T., Cáceres, J.F., Clement, J.Q., Krainer, A.R., Wilkinson, M.F., and Spector, D.L. (1998). Serine Phosphorylation of SR Proteins Is Required for Their Recruitment to Sites of Transcription In Vivo. The Journal of Cell Biology *143*, 297–307.

Moyer, S.E., Lewis, P.W., and Botchan, M.R. (2006). Isolation of the Cdc45/Mcm2–7/GINS (CMG) complex, a candidate for the eukaryotic DNA replication fork helicase. Proc Natl Acad Sci U S A *103*, 10236–10241.

Näär, A.M., Taatjes, D.J., Zhai, W., Nogales, E., and Tjian, R. (2002). Human CRSP interacts with RNA polymerase II CTD and adopts a specific CTD-bound conformation. Genes Dev. *16*, 1339–1344.

Nagalakshmi, U., Wang, Z., Waern, K., Shou, C., Raha, D., Gerstein, M., and Snyder, M. (2008). The Transcriptional Landscape of the Yeast Genome Defined by RNA Sequencing. Science *320*, 1344–1349.

Nakama, M., Kawakami, K., Kajitani, T., Urano, T., and Murakami, Y. (2012). DNA–RNA hybrid formation mediates RNAi-directed heterochromatin formation. Genes to Cells *17*, 218–233.

Neelsen, K.J., and Lopes, M. (2015). Replication fork reversal in eukaryotes: from dead end to dynamic response. Nat Rev Mol Cell Biol *16*, 207–220.

Novoyatleva, T., Heinrich, B., Tang, Y., Benderska, N., Butchbach, M.E.R., Lorson, C.L., Lorson, M.A., Ben-Dov, C., Fehlbaum, P., Bracco, L., et al. (2008). Protein phosphatase 1 binds to the RNA recognition motif of several splicing factors and regulates alternative pre-mRNA processing. Hum Mol Genet *17*, 52–70.

Okazaki, R., Okazaki, T., Sakabe, K., Sugimoto, K., and Sugino, A. (1968). Mechanism of DNA chain growth. I. Possible discontinuity and unusual secondary structure of newly synthesized chains. Proc Natl Acad Sci U S A *59*, 598–605.

Park, P.J. (2009). ChIP–seq: advantages and challenges of a maturing technology. Nat Rev Genet *10*, 669–680.

Parker, C.S., and Topol, J. (1984). A Drosophila RNA polymerase II transcription factor contains a promoter-region-specific DNA-binding activity. Cell *36*, 357–369.

Pedersen, J.M., Fredsoe, J., Roedgaard, M., Andreasen, L., Mundbjerg, K., Kruhøffer, M., Brinch, M., Schierup, M.H., Bjergbaek, L., and Andersen, A.H. (2012). DNA Topoisomerases Maintain Promoters in a State Competent for Transcriptional Activation in Saccharomyces cerevisiae. PLOS Genetics *8*, e1003128.

Pei, Y., and Shuman, S. (2002). Interactions between Fission Yeast mRNA Capping Enzymes and Elongation Factor Spt5. J. Biol. Chem. *277*, 19639–19648.

Petryk, N., Kahli, M., d'Aubenton-Carafa, Y., Jaszczyszyn, Y., Shen, Y., Silvain, M., Thermes, C., Chen, C.-L., and Hyrien, O. (2016). Replication landscape of the human genome. Nature Communications *7*, ncomms10208.

Pommier, Y. (2006). Topoisomerase I inhibitors: camptothecins and beyond. Nat Rev Cancer *6*, 789–802.

Pommier, Y., Leo, E., Zhang, H., and Marchand, C. (2010). DNA Topoisomerases and Their Poisoning by Anticancer and Antibacterial Drugs. Chemistry & Biology *17*, 421–433.

Pommier, Y., Sun, Y., Huang, S.N., and Nitiss, J.L. (2016). Roles of eukaryotic topoisomerases in transcription, replication and genomic stability. Nat Rev Mol Cell Biol *17*, 703–721.

Porter, S.E., and Champoux, J.J. (1989). The basis for camptothecin enhancement of DNA breakage by eukaryotic topoisomerase I. Nucleic Acids Res *17*, 8521–8532.

Prado, F., and Aguilera, A. (2005). Impairment of replication fork progression mediates RNA polII transcription-associated recombination. The EMBO Journal *24*, 1267–1276.

Proudfoot, N.J. (2016). Transcriptional termination in mammals: Stopping the RNA polymerase II juggernaut. Science *352*, aad9926.

Proudfoot, N.J., Furger, A., and Dye, M.J. (2002). Integrating mRNA Processing with Transcription. Cell *108*, 501–512.

Qiu, J., Qian, Y., Frank, P., Wintersberger, U., and Shen, B. (1999). Saccharomyces cerevisiae RNase H(35) Functions in RNA Primer Removal during Lagging-Strand DNA Synthesis, Most Efficiently in Cooperation with Rad27 Nuclease. Mol Cell Biol *19*, 8361–8371.

Quinlan, A.R. (2002). BEDTools: The Swiss-Army Tool for Genome Feature Analysis. In Current Protocols in Bioinformatics, (John Wiley & Sons, Inc.), p.

Ragland, R.L., Patel, S., Rivard, R.S., Smith, K., Peters, A.A., Bielinsky, A.-K., and Brown, E.J. (2013). RNF4 and PLK1 are required for replication fork collapse in ATR-deficient cells. Genes Dev. *27*, 2259–2273.

Ramírez, F., Dündar, F., Diehl, S., Grüning, B.A., and Manke, T. (2014). deepTools: a flexible platform for exploring deep-sequencing data. Nucleic Acids Res *42*, W187–W191.

Rampakakis, E., Gkogkas, C., Di Paola, D., and Zannis-Hadjopoulos, M. (2010). Replication initiation and DNA topology: The twisted life of the origin. J. Cell. Biochem. *110*, 35–43.

Rapaport, F., Khanin, R., Liang, Y., Pirun, M., Krek, A., Zumbo, P., Mason, C.E., Socci, N.D., and Betel, D. (2013). Comprehensive evaluation of differential gene expression analysis methods for RNA-seq data. Genome Biology *14*, 3158.

Reese, J.C. (2003). Basal transcription factors. Current Opinion in Genetics & Development *13*, 114–118.

Reuter, J.A., Spacek, D.V., and Snyder, M.P. (2015). High-Throughput Sequencing Technologies. Molecular Cell *58*, 586–597.

Robertson, G., Hirst, M., Bainbridge, M., Bilenky, M., Zhao, Y., Zeng, T., Euskirchen, G., Bernier, B., Varhol, R., Delaney, A., et al. (2007). Genome-wide profiles of STAT1 DNA association using chromatin immunoprecipitation and massively parallel sequencing. Nat Meth *4*, 651–657.

Robinson, M.D., McCarthy, D.J., and Smyth, G.K. (2010). edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics *26*, 139–140.

Robles, J.A., Qureshi, S.E., Stephen, S.J., Wilson, S.R., Burden, C.J., and Taylor, J.M. (2012). Efficient experimental design and analysis strategies for the detection of differential expression using RNA-Sequencing. BMC Genomics *13*, 484.

Roedgaard, M., Fredsoe, J., Pedersen, J.M., Bjergbaek, L., and Andersen, A.H. (2015). DNA Topoisomerases Are Required for Preinitiation Complex Assembly during GAL Gene Activation. PLoS One *10*.

Rogakou, E.P., Boon, C., Redon, C., and Bonner, W.M. (1999). Megabase Chromatin Domains Involved in DNA Double-Strand Breaks in Vivo. J Cell Biol *146*, 905–916.

Roy, D., and Lieber, M.R. (2009). G Clustering Is Important for the Initiation of Transcription-Induced R-Loops In Vitro, whereas High G Density without Clustering Is Sufficient Thereafter. Mol. Cell. Biol. *29*, 3124–3133.

Roy, D., Yu, K., and Lieber, M.R. (2008). Mechanism of R-Loop Formation at Immunoglobulin Class Switch Sequences. Mol Cell Biol *28*, 50–60.

Roy, D., Zhang, Z., Lu, Z., Hsieh, C.-L., and Lieber, M.R. (2010). Competition between the RNA Transcript and the Nontemplate DNA Strand during R-Loop Formation In Vitro: a Nick Can Serve as a Strong R-Loop Initiation Site. Mol. Cell. Biol. *30*, 146–159.

Sanford, J.R., Gray, N.K., Beckmann, K., and Cáceres, J.F. (2004). A novel role for shuttling SR proteins in mRNA translation. Genes Dev. *18*, 755–768.

Sanger, F., Nicklen, S., and Coulson, A.R. (1977). DNA sequencing with chain-terminating inhibitors. Proc Natl Acad Sci U S A *74*, 5463–5467.

Santos-Pereira, J.M., and Aguilera, A. (2015). R loops: new modulators of genome dynamics and function. Nat Rev Genet *16*, 583–597.

Sanz, L.A., Hartono, S.R., Lim, Y.W., Steyaert, S., Rajpurkar, A., Ginno, P.A., Xu, X., and Chédin, F. (2016). Prevalent, dynamic, and conserved R-loop structures associate with specific epigenomic signatures in mammals. Mol Cell *63*, 167–178.

Schlacher, K., Christ, N., Siaud, N., Egashira, A., Wu, H., and Jasin, M. (2011). Double-Strand Break Repair-Independent Role for BRCA2 in Blocking Stalled Replication Fork Degradation by MRE11. Cell *145*, 529–542.

Schlacher, K., Wu, H., and Jasin, M. (2012). A Distinct Replication Fork Protection Pathway Connects Fanconi Anemia Tumor Suppressors to RAD51-BRCA1/2. Cancer Cell *22*, 106–116.

Seyednasrollah, F., Laiho, A., and Elo, L.L. (2015). Comparison of software packages for detecting differential expression in RNA-seq studies. Brief Bioinform *16*, 59–70.

Shen, H., and Green, M.R. (2004). A Pathway of Sequential Arginine-Serine-Rich Domain-Splicing Signal Interactions during Mammalian Spliceosome Assembly. Molecular Cell *16*, 363–373.

Shen, H., Kan, J.L.C., and Green, M.R. (2004). Arginine-Serine-Rich Domains Bound at Splicing Enhancers Contact the Branchpoint to Promote Prespliceosome Assembly. Molecular Cell *13*, 367–376.

Shin, H., Liu, T., Manrai, A.K., and Liu, X.S. (2009). CEAS: cis-regulatory element annotation system. Bioinformatics *25*, 2605–2606.

Shykind, B.M., Kim, J., Stewart, L., Champoux, J.J., and Sharp, P.A. (1997). Topoisomerase I enhances TFIID-TFIIA complex assembly during activation of transcription. Genes Dev. *11*, 397–407.

Sirbu, B.M., McDonald, W.H., Dungrawala, H., Badu-Nkansah, A., Kavanaugh, G.M., Chen, Y., Tabb, D.L., and Cortez, D. (2013). Identification of Proteins at Active, Stalled, and Collapsed Replication Forks Using Isolation of Proteins on Nascent DNA (iPOND) Coupled with Mass Spectrometry. J. Biol. Chem. *288*, 31458–31467.

Skourti-Stathaki, K., and Proudfoot, N.J. (2014). A double-edged sword: R loops as threats to genome integrity and powerful regulators of gene expression. Genes Dev *28*, 1384–1396.

Skourti-Stathaki, K., Proudfoot, N.J., and Gromak, N. (2011). Human Senataxin Resolves RNA/DNA Hybrids Formed at Transcriptional Pause Sites to Promote Xrn2-Dependent Termination. Molecular Cell *42*, 794–805.

Skourti-Stathaki, K., Kamieniarz-Gdula, K., and Proudfoot, N.J. (2014). R-loops induce repressive chromatin marks over mammalian gene terminators. Nature *516*, 436–439.

Smith, D.I., Zhu, Y., McAvoy, S., and Kuhn, R. (2006). Common fragile sites, extremely large genes, neural development and cancer. Cancer Letters *232*, 48–57.

Song, Q., and Smith, A.D. (2011). Identifying dispersed epigenomic domains from ChIP-Seq data. Bioinformatics *27*, 870–871.

Sordet, O., Larochelle, S., Nicolas, E., Stevens, E.V., Zhang, C., Shokat, K.M., Fisher, R.P., and Pommier, Y. (2008). RNA polymerase II is Hyperphosphorylated in Response to Topoisomerase I-DNA Cleavage Complexes and is Associated with Transcription- and BRCA1-Dependent Degradation of Topoisomerase I. J Mol Biol *381*, 540–549.

Sordet, O., Redon, C.E., Guirouilh-Barbat, J., Smith, S., Solier, S., Douarre, C., Conti, C., Nakamura, A.J., Das, B.B., Nicolas, E., et al. (2009). Ataxia telangiectasia mutated activation by transcription- and topoisomerase I-induced DNA double-strand breaks. EMBO Reports *10*, 887–893.

Sordet, O., Nakamura, A.J., Redon, C.E., and Pommier, Y. (2010). DNA double-strand breaks and ATM activation by transcription-blocking DNA lesions. Cell Cycle *9*, 274–278.

Sørensen, C.S., and Syljuåsen, R.G. (2012). Safeguarding genome integrity: the checkpoint kinases ATR, CHK1 and WEE1 restrain CDK activity during normal DNA replication. Nucleic Acids Res *40*, 477–486.

Sørensen, C.S., Syljuåsen, R.G., Falck, J., Schroeder, T., Rönnstrand, L., Khanna, K.K., Zhou, B.-B., Bartek, J., and Lukas, J. (2003). Chk1 regulates the S phase checkpoint by coupling the physiological turnover and ionizing radiation-induced accelerated proteolysis of Cdc25A. Cancer Cell *3*, 247–258.

Sperling, A.S., Jeong, K.S., Kitada, T., and Grunstein, M. (2011). Topoisomerase II binds nucleosome-free DNA and acts redundantly with topoisomerase I to enhance recruitment of RNA Pol II in budding yeast. Proc Natl Acad Sci U S A *108*, 12693–12698.

Spyrou, C., Stark, R., Lynch, A.G., and Tavaré, S. (2009). BayesPeak: Bayesian analysis of ChIP-seq data. BMC Bioinformatics *10*, 299.

Srivatsan, A., Tehranchi, A., MacAlpine, D.M., and Wang, J.D. (2010). Co-Orientation of Replication and Transcription Preserves Genome Integrity. PLOS Genetics *6*, e1000810.

Steinhauser, S., Kurzawa, N., Eils, R., and Herrmann, C. (2016). A comprehensive comparison of tools for differential ChIP-seq analysis. Brief Bioinform *17*, 953–966.

Stith, C.M., Sterling, J., Resnick, M.A., Gordenin, D.A., and Burgers, P.M. (2008). Flexibility of Eukaryotic Okazaki Fragment Maturation through Regulated Strand Displacement Synthesis. J. Biol. Chem. *283*, 34129–34140.

Strumberg, D., Pilon, A.A., Smith, M., Hickey, R., Malkas, L., and Pommier, Y. (2000). Conversion of Topoisomerase I Cleavage Complexes on the Leading Strand of

Ribosomal DNA into 5′-Phosphorylated DNA Double-Strand Breaks by Replication Runoff. Mol Cell Biol *20*, 3977–3987.

Sun, Q., Csorba, T., Skourti-Stathaki, K., Proudfoot, N.J., and Dean, C. (2013). R-Loop Stabilization Represses Antisense Transcription at the Arabidopsis FLC Locus. Science *340*, 619–621.

Svejstrup, J.Q. (2002). Chromatin elongation factors. Current Opinion in Genetics & Development *12*, 156–161.

Symeonidou, I.-E., Taraviras, S., and Lygerou, Z. (2012). Control over DNA replication in time and space. FEBS Letters 586, 2803–2812.

Tate, P.H., and Bird, A.P. (1993). Effects of DNA methylation on DNA-binding proteins and gene expression. Current Opinion in Genetics & Development *3*, 226–231.

Técher, H., Koundrioukoff, S., Nicolas, A., and Debatisse, M. (2017). The impact of replication stress on replication dynamics and DNA damage in vertebrate cells. Nat Rev Genet *18*, 535–550.

The ENCODE Project Consortium, T.E.P. (2012). An integrated encyclopedia of DNA elements in the human genome. Nature *489*, 57–74.

Thomas, M., White, R.L., and Davis, R.W. (1976). Hybridization of RNA to double-stranded DNA: formation of R-loops. Proc Natl Acad Sci U S A *73*, 2294–2298.

Tran, D.P., Kim, S.J., Park, N.J., Jew, T.M., and Martinson, H.G. (2001). Mechanism of Poly(A) Signal Transduction to RNA Polymerase II In Vitro. Mol Cell Biol *21*, 7495–7508.

Trapnell, C., Roberts, A., Goff, L., Pertea, G., Kim, D., Kelley, D.R., Pimentel, H., Salzberg, S.L., Rinn, J.L., and Pachter, L. (2012). Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks. Nat. Protocols *7*, 562–578.

Trenz, K., Smith, E., Smith, S., and Costanzo, V. (2006). ATM and ATR promote Mre11 dependent restart of collapsed replication forks and prevent accumulation of DNA breaks. The EMBO Journal *25*, 1764–1774.

Tuduri, S., Crabbé, L., Conti, C., Tourrière, H., Holtgreve-Grez, H., Jauch, A., Pantesco, V., De Vos, J., Thomas, A., Theillet, C., et al. (2009). Topoisomerase I suppresses genomic instability by preventing interference between replication and transcription. Nat Cell Biol *11*, 1315–1324.

Vassin, V.M., Anantha, R.W., Sokolova, E., Kanner, S., and Borowiec, J.A. (2009). Human RPA phosphorylation by ATR stimulates DNA synthesis and prevents ssDNA accumulation during DNA-replication stress. Journal of Cell Science *122*, 4070–4080.

Vieira, K.F., Levings, P.P., Hill, M.A., Crusselle, V.J., Kang, S.-H.L., Engel, J.D., and Bungert, J. (2004). Recruitment of Transcription Complexes to the β-Globin Gene Locus in Vivo and in Vitro. J. Biol. Chem. *279*, 50350–50357.

Vogelstein, B., Lane, D., and Levine, A.J. (2000). Surfing the p53 network. Nature *408*, 307–310.

Vos, S.M., Tretter, E.M., Schmidt, B.H., and Berger, J.M. (2011). All tangled up: how cells direct, manage and exploit topoisomerase function. Nat Rev Mol Cell Biol *12*, 827–841.

Wada, T., Takagi, T., Yamaguchi, Y., Ferdous, A., Imai, T., Hirose, S., Sugimoto, S., Yano, K., Hartzog, G.A., Winston, F., et al. (1998). DSIF, a novel transcription elongation factor that regulates RNA polymerase II processivity, is composed of human Spt4 and Spt5 homologs. Genes Dev. *12*, 343–356.

Wang, J.C. (2002). Cellular roles of DNA topoisomerases: a molecular perspective. Nat Rev Mol Cell Biol *3*, 430–440.

Wang, E.T., Sandberg, R., Luo, S., Khrebtukova, I., Zhang, L., Mayr, C., Kingsmore, S.F., Schroth, G.P., and Burge, C.B. (2008). Alternative isoform regulation in human tissue transcriptomes. Nature *456*, 470–476.

Wang, Z., Gerstein, M., and Snyder, M. (2009). RNA-Seq: a revolutionary tool for transcriptomics. Nat Rev Genet *10*, 57–63.

Watson, J.D., and Crick, F.H. (1953). The structure of DNA. Cold Spring Harb. Symp. Quant. Biol. *18*, 123–131.

Wei, X., Samarabandu, J., Devdhar, R.S., Siegel, A.J., Acharya, R., and Berezney, R. (1998). Segregation of Transcription and Replication Sites Into Higher Order Domains. Science *281*, 1502–1505.

Weiss, A.S., Hariharan, I.K., and Wake, R.G. (1981). Analysis of the terminus region of the Bacillus subtilis chromosome. Nature *293*, 673–675.

Wen, Y., and Shatkin, A.J. (1999). Transcription elongation factor hSPT5 stimulates mRNA capping. Genes Dev. *13*, 1774–1779.

Whitby, M.C., Osman, F., and Dixon, J. (2003). Cleavage of Model Replication Forks by Fission Yeast Mus81-Eme1 and Budding Yeast Mus81-Mms4. J. Biol. Chem. *278*, 6928–6935.

White, R.L., and Hogness, D.S. (1977). R loop mapping of the 18S and 28S sequences in the long and short repeating units of drosophila melanogaster rDNA. Cell *10*, 177–192.

Whitington, T., Frith, M.C., Johnson, J., and Bailey, T.L. (2011). Inferring transcription factor complexes from ChIP-seq data. Nucleic Acids Res *39*, e98–e98.

Wilbanks, E.G., and Facciotti, M.T. (2010). Evaluation of Algorithm Performance in ChIP-Seq Peak Detection. PLOS ONE *5*, e11471.

Wimberly, H., Shee, C., Thornton, P.C., Sivaramakrishnan, P., Rosenberg, S.M., and Hastings, P.J. (2013). R-loops and nicks initiate DNA breakage and genome instability in non-growing Escherichia coli. Nat Commun *4*.

Woodfine, K., Fiegler, H., Beare, D.M., Collins, J.E., McCann, O.T., Young, B.D., Debernardi, S., Mott, R., Dunham, I., and Carter, N.P. (2004). Replication timing of the human genome. Hum Mol Genet *13*, 191–202.

Wu, C.-H., Yamaguchi, Y., Benjamin, L.R., Horvat-Gordon, M., Washinsky, J., Enerly, E., Larsson, J., Lambertsson, A., Handa, H., and Gilmour, D. (2003). NELF and DSIF cause promoter proximal pausing on the hsp70 promoter in Drosophila. Genes Dev *17*, 1402–1414.

Xu, S., Grullon, S., Ge, K., and Peng, W. (2014). Spatial Clustering for Identification of ChIP-Enriched Regions (SICER) to Map Regions of Histone Methylation Patterns in Embryonic Stem Cells. Methods Mol Biol *1150*, 97–111.

Yamaguchi, Y., Takagi, T., Wada, T., Yano, K., Furuya, A., Sugimoto, S., Hasegawa, J., and Handa, H. (1999). NELF, a Multisubunit Complex Containing RD, Cooperates with DSIF to Repress RNA Polymerase II Elongation. Cell *97*, 41–51.

Yang, Y., McBride, K.M., Hensley, S., Lu, Y., Chedin, F., and Bedford, M.T. (2014). Arginine Methylation Facilitates the Recruitment of TOP3B to Chromatin to Prevent R Loop Accumulation. Molecular Cell *53*, 484–497.

Yu, K., Chedin, F., Hsieh, C.-L., Wilson, T.E., and Lieber, M.R. (2003). R-loops at immunoglobulin class switch regions in the chromosomes of stimulated B cells. Nat Immunol *4*, 442–451.

Zeman, M.K., and Cimprich, K.A. (2014). Causes and consequences of replication stress. Nat Cell Biol *16*, 2–9.

Zhang, H., Rigo, F., and Martinson, H.G. (2015). Poly(A) Signal-Dependent Transcription Termination Occurs through a Conformational Change Mechanism that Does Not Require Cleavage at the Poly(A) Site. Molecular Cell *59*, 437–448.

Zhang, Y., Liu, T., Meyer, C.A., Eeckhoute, J., Johnson, D.S., Bernstein, B.E., Nusbaum, C., Myers, R.M., Brown, M., Li, W., et al. (2008). Model-based Analysis of ChIP-Seq (MACS). Genome Biol *9*, R137.

Zheng, L., and Shen, B. (2011). Okazaki fragment maturation: nucleases take centre stage. J Mol Cell Biol *3*, 23–30.

Résumé

L'activation d'oncogènes entraine une prolifération aberrante des cellules, un stress réplicatif et des cassures de l'ADN. Un lien a été établi entre l'instabilité génomique résultant des cassures et l'inhibition de checkpoints entrainant l'accumulation de mutations et finalement le cancer (Halazonetis et al. 2008). Cependant, les mécanismes liant ces différents évènements n'ont pas encore été caractérisés. Notre hypothèse est que la prolifération incontrôlée des cellules augmente les incidents dus aux conflits entre les polymérases responsables de la réplication et celles responsables de la transcription. Lors de la rencontre des deux polymérases, l'accumulation de surenroulements positifs de l'ADN induit un blocage des fourches de réplication. Ceci crée des zones de fragilité, notamment dues à l'exposition d'ADN simple brin, et pourrait être à l'origine des cassures observées chez les cellules tumorales. Pour valider cette hypothèse, les biologistes de l'équipe ont étudié plusieurs lignées de cellules HeLa dans lesquelles les conflits réplication-transcription sont augmentés et j'ai réalisé l'analyse bioinformatique des approches génomiques suivantes :

- DRIP-seq pour la détection des R-loops, une structure double brin hybride ADN/ARN qui se forme lors de la transcription, exposant ainsi un brin d'ADN simple brin.
- ChIP-seq de γ-H2AX, une marque d'histone indiquant les cassures de l'ADN.
- ChIP-seq de phospho-RPA (S33), un substrat de la kinase ATR au niveau des fourches bloquées.

Pour chaque expérience, nous avons utilisé une lignée contrôle et deux lignées dans lesquelles TOP1 et ASF/SF2 sont appauvries avec un shRNA inductible (shTOP1 et shASF). La Topoisomérase I (TOP1) est une enzyme qui relaxe les surenroulements de l'ADN. Le complexe ASF/SF2 est un facteur d'épissage responsable entre autres de l'assemblage des mRNP (ribonucleoprotein particles) au moment de la transcription, qui limite la formation des R-loops. L'analyse bioinformatique de ces données, ainsi que d'autres données de la littérature, m'a permis d'identifier des régions à risque du génome, localisées en aval de gènes fortement transcrits et répliqués précocement en phase S par des fourches progressant en sens opposé à la transcription. J'ai également observé que les gènes impliqués dans le cancer sont surreprésentés dans ces régions à risque.

Abstract

Oncogenes activation promotes aberrant cell proliferation, increasing replication stress and DNA damage. It has been proposed that genomic instability leads to checkpoints inhibition and promotes cancer development (Halazonetis et al. 2008). However, the link between aberrant proliferation, replication stress and DNA breaks is still unclear. We hypothesized that aberrant proliferation leads to more incident due to DNA and RNA polymerases encounter and stalling. When the two polymerases encounter, the accumulation of positive-supercoiled DNA between two polymerases induces fork stalling, resulting in the formation of fragile structures such as single-stranded DNA (ssDNA). These ssDNAs formed at stalled forks could be a source for DNA breaks, promoting the development of cancer cells. To validate this hypothesis, biologists from our team have worked on HeLa cell lines with increased replication-transcription conflicts. I perform the bioinformatics analysis of the following genomic data:

- DRIP-seq: R-Loops positioning on genome using immunoprecipitation on DNA/RNA hybrids.
- γ-H2AX ChIP-Seq: Gamma-H2AX is an histone mark found at DNA breaks.
- pRPA ChIP-Seq : Positioning of stalled forks using the substrate of ATR kinase, phospho-RPA (S33) as a marker.

Each data was produced on control cells and two cell lines where TOP1 and ASF/SF2 were depleted by as inducible shRNA (shTOP1 and shASF). Topoisomerase 1 is a topological enzyme that unwinds DNA when supercoiling accumulates. ASF/SF2 is part of the splicing complexes that processes mRNP (messenger ribonucleoprotein particles) to prevent the accumulation of R-loops during transcription. Using these data and others from literature, I determined that regions having higher risk to induce replication stress are located downstream of highly transcribed and early replicated genes, and preferentially with head-on collision between DNA and RNA polymerases. I also revealed that cancer-related genes are enriched in these regions of the genome.